

RADC-TR-89-215
Interim Report
January 1990



AD-A219 928

KNOWLEDGE-BASED LOGISTICS PLANNING AND ITS APPLICATION IN MANUFACTURING AND STRATEGIC PLANNING

Carnegie Mellon University

Mark S. Fox and Katla P. Sycara

Sponsored by
Defense Advanced Research Projects Agency
ARPA Order No. 6129

DTIC
ELECTE
MAR 30 1990
S E D

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

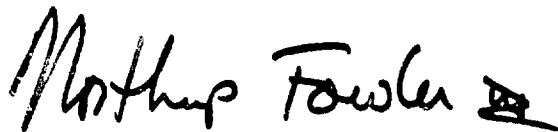
Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

90 03 29 131

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Services (NTIS) At NTIS it will be releasable to the general public, including foreign nations.

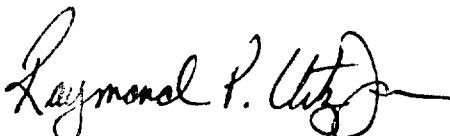
RADC-TR-89-215 has been reviewed and is approved for publication.

APPROVED:



NORTHROP FOWLER III
Project Engineer

APPROVED:



RAYMOND P. URTZ, JR
Technical Director
Directorate of Command & Control

FOR THE COMMANDER:



IGOR G. PLONISCH
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COES) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

KNOWLEDGE-BASED LOGISTICS PLANNING AND ITS APPLICATION
IN MANUFACTURING AND STRATEGIC PLANNING

Mark S. Fox
Katia P. Sycara

Contractor: Carnegie Mellon University
Contract Number: F30602-88-C-0001
Effective Date of Contract: 30 November 1987
Contract Expiration Date: 30 November 1990
Short Title of Work: Knowledge-Based Logistics Planning and
Its Application in Manufacturing and Strategic Planning
Program Code Number: 9E20
Period of Work Covered: Nov 87 - Nov 88

Principal Investigator: Mark S. Fox
Phone: (412) 268-3832

RADC Project Engineer: Northrup Fowler III
Phone: (315) 330-7794

Approved for public release; distribution unlimited.

This research was supported by the Defense Advanced
Research Projects Agency of the Department of Defense
and was monitored by Northrup Fowler III, RADC (COES),
Griffiss AFB NY 13441-5700 under Contract F30602-88-
C-0001.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS N/A		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A			5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-89-215		
6a. NAME OF PERFORMING ORGANIZATION Carnegie Mellon University		6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COES)		
6c. ADDRESS (City, State, and ZIP Code) 500 Forbes Ave Pittsburgh PA 15213			7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Advanced Research Projects Agency		8b. OFFICE SYMBOL (If applicable) ISTO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-88-C-0001		
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd Arlington VA 22209-2308			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO. 62301E	PROJECT NO. F129	TASK NO. 00
11. TITLE (Include Security Classification) KNOWLEDGE-BASED LOGISTICS PLANNING AND ITS APPLICATION IN MANUFACTURING AND STRATEGIC PLANNING					
12. PERSONAL AUTHOR(S) Mark S. Fox, Katia P. Sycara					
13a. TYPE OF REPORT Interim		13b. TIME COVERED FROM Nov 87 TO Nov 88		14. DATE OF REPORT (Year, Month, Day) January 1990	
15. PAGE COUNT 130					
16. SUPPLEMENTARY NOTATION N/A					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Artificial intelligence, Constraint based reasoning, Planning, Manufacturing, Logistics		
FIELD	GROUP	SUB-GROUP			
12	05				
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This report summarizes the activities and results of the first year's effort on the project that started at the end of November 1987. This effort concentrated on investigating the constraint-directed reasoning behavior of a single agent. The report is divided into three chapters. The first chapter presents the overall problem solving framework, an integration of constraint satisfaction and heuristic search. The second chapter presents the propagation of preferential constraints pertaining to temporal relations and resource capacities. The third chapter describes the focus of attention mechanism of a constraint-directed scheduler and some preliminary experimental results. The fourth chapter presents the representational framework for the concepts used in the research.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Northrup Fowler III			22b. TELEPHONE (Include Area Code) (315) 330-7794		22c. OFFICE SYMBOL RADC (COES)

Table of Contents

INTRODUCTION	1
	2
CHAPTER 1: Constraint Satisfaction and Heuristic Search	
1.1 Introduction	2
1.2 Problem Space Topology	3
1.3 Problem Space Textures	5
1.4 CHS Problem Solving Process	8
CHAPTER 2: Propagating Temporal and Capacity Preferences	
2.1 Introduction	9
2.1.1 The Issue	9
2.1.2 Formalization of the Scheduling Problem	10
2.1.3 An Example	12
2.1.4 Organization of the Paper	15
2.2 Related Work	16
2.3 A Probabilistic Framework for Preference Propagation	19
2.3.1 An Overview of the Approach	19
2.3.2 Building A Priori Probability Distributions Based on Local A Priori Preferences	21
2.4 Propagating Start Time and Duration Distributions in a TCG	23
2.4.1 Preliminary Remarks	23
2.4.2 Propagation in an Acyclic TCG with Fixed-duration Activities	25
2.4.2.1 $C_i^0: I_0$ MEETS I_i^0	27
2.4.2.2 $C_i^0: I_0$ is BEFORE I_i^0	28
2.4.2.3 Example	29
2.4.3 Propagation in an Acyclic TCG with Variable-duration Activities	30
2.4.3.1 $C_i^0: I_0$ MEETS I_i^0	31
2.4.3.2 $C_i^0: I_0$ BEFORE I_i^0	31
2.4.4 Relaxing the Acyclicity Assumption	32
2.4.5 Propagation in a TCG with Explicit Disjunctions	40
2.4.6 Result Interpretation	41
2.5 Resource Demand Densities	42
2.6 Discussion	45
2.6.1 Time Complexity	45
2.6.2 Expressiveness of the Model	45
2.6.3 Possible Improvements	46
2.6.3.1 Iterative and Hierarchical Preference Propagation	46
2.6.3.2 Activity Criticality	46
2.6.3.3 Value Goodness	47
2.7 Summary and Concluding Remarks	47
Appendix1: A Posteriori Start Time and Duration Distributions	49
0.1 Acyclic TCG with fixed-duration activities	49
0.1.1 $C_i^0: I_0$ MEETS I_i^0	49
0.1.2 $C_i^0: I_0$ MET-BY I_i^0	49

0.1.3 C_i^0 : I_0 BEFORE I_i^0	49
0.1.4 C_i^0 : I_0 AFTER I_i^0	49
0.1.5 C_i^0 : I_0 DURING I_i^0	50
0.1.6 C_i^0 : I_0 CONTAINS I_i^0	50
0.1.7 C_i^0 : I_0 STARTS I_i^0	50
0.1.8 C_i^0 : I_0 STARTED-BY I_i^0	50
0.1.9 C_i^0 : I_0 FINISHES I_i^0	50
0.1.10 C_i^0 : I_0 FINISHED-BY I_i^0	50
0.1.11 C_i^0 : I_0 OVERLAPS I_i^0	50
0.1.12 C_i^0 : I_0 OVERLAPPED-BY I_i^0	51
0.1.13 C_i^0 : I_0 EQUALS I_i^0	51
0.2 Acyclic TCG with variable-duration activities	51
0.2.1 C_i^0 : I_0 MEETS I_i^0	51
0.2.2 C_i^0 : I_0 MET-BY I_i^0	51
0.2.3 C_i^0 : I_0 BEFORE I_i^0	52
0.2.4 C_i^0 : I_0 AFTER I_i^0	52
0.2.5 C_i^0 : I_0 DURING I_i^0	52
0.2.6 C_i^0 : I_0 CONTAINS I_i^0	52
0.2.7 C_i^0 : I_0 STARTS I_i^0	52
0.2.8 C_i^0 : I_0 STARTED-BY I_i^0	52
0.2.9 C_i^0 : I_0 FINISHES I_i^0	53
0.2.10 C_i^0 : I_0 FINISHED-BY I_i^0	53
0.2.11 C_i^0 : I_0 OVERLAPS I_i^0	53
0.2.12 C_i^0 : I_0 OVERLAPPED-BY I_i^0	53
0.2.13 C_i^0 : I_0 EQUALS I_i^0	54
Appendix2: Activity Individual Demand Densities	55
0.1 Notations	55
0.2 Resource Demand Densities Produced by Fixed-Duration Activities	55
0.3 Resource Demand Densities Produced by Variable-Duration Activities	57

CHAPTER 3: Activity-Based Scheduling

3.1 Introduction	58
3.2 The Model	59
3.3 The Approach	60
3.3.1 An Activity-based Scheduler	60
3.3.2 A Probabilistic Framework to Account for Constraint Interactions	60
3.4 ARR: A Variable Ordering Heuristic Based on Activity Resource Reliance	62
3.5 Three Value Ordering Heuristics	63
3.5.1 LCV: A Least Constraining Value Ordering Heuristic	63
3.5.2 HC: A Hill-Climbing Value Ordering Heuristic	64
3.5.3 INT: An Intermediate Value Ordering Heuristic	64
3.6 Preliminary Experimental Results	65

3.7 Discussion	66
3.8 Concluding Remarks	67

CHAPTER 4: Representation

4.1 Introduction	68
4.2 Activities	68
4.3 States	72
4.4 Resources	75
4.5 Production Units	77
4.6 Constraints	78
4.6.1 Representing Constraints	78
4.6.2 Representing temporal relations	84

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Code	
Dist	Avail and/or Special
A-1	



List of Figures

Figure 2-1:	Allen's 13 temporal relation constraints	11
Figure 2-2:	TCG for a two order scheduling problem	13
Figure 2-3:	Start time utility functions	14
Figure 2-4:	Gantt chart for a schedule of order1 obtained with scenario1. The global start time utility has been obtained by adding the start time utility of the five activities.	15
Figure 2-5:	Gantt chart for a schedule of order1 obtained with scenario2. The global start time utility has been obtained by adding the start time utility of the five activities.	15
Figure 2-6:	An inconsistent CSP with a consistent TCG	17
Figure 2-7:	A utility interpretation of DEVISER's start time windows	18
Figure 2-8:	Rit's generalized window	18
Figure 2-9:	A priori probability density $P(x)$ for a variable x with utility function $u(x)$. This is an example with a single optimal value.	22
Figure 2-10:	Example of a TCG with no explicit disjunctions	24
Figure 2-11:	Example of a TCG with explicit disjunctions	24
Figure 2-12:	An Acyclic TCG. The time interval I_0 is related to $I_1^0, I_2^0, \dots,$ $I_{p_0}^0$ by respectively $C_1^0, C_2^0, \dots, C_{p_0}^0$.	26
Figure 2-13:	I_0 MEETS I_i^0	28
Figure 2-14:	I_0 BEFORE I_i^0	28
Figure 2-15:	A posteriori start time densities for order ₂	30
Figure 2-16:	Illustration of Fubini's Theorem in a TCG with 3 Time Periods	33
Figure 2-17:	A TCG with 3 Time Periods	35
Figure 2-18:	Procedure to express $P(C_1, \dots, C_m st_0 = t \& du_0 = d)$ as an iterated integral	37
Figure 2-19:	Main steps involved in the construction of (15). Notice that the α expressions have been omitted as they trivially evaluate to 1.	38
Figure 2-20:	A posteriori start time densities for order ₁	39
Figure 2-21:	A TCG with explicit disjunctions	40
Figure 2-22:	$R_1, R_2,$ and R_3 's aggregate demand densities	43
Figure 2-23:	Contributions of $A_3, A_4,$ and A_7 to R_2 's aggregate demand density	44
Figure 3-1:	Average search efficiencies and schedule values for 5 combinations of variable and value ordering heuristics run on a preliminary set of 38 scheduling problems. The standard deviations appear between parentheses.	65
Figure 4-1:	Relations representing elaboration of activities	69
Figure 4-2:	Precedence relations between activities	69
Figure 4-3:	Aggregate Activity	70
Figure 4-4:	A Process Plan	70
Figure 4-5:	The relation has-subactivity	70
Figure 4-6:	The relation subactivity-of	71
Figure 4-7:	The relation required-resource	71

Figure 4-8:	The relation resource-for	71
Figure 4-9:	Representation of an activity	72
Figure 4-10:	Representation of an operation	72
Figure 4-11:	Relations defining enablement of activities	73
Figure 4-12:	Relations defining activity postconditions	73
Figure 4-13:	Relations defining state aggregation	74
Figure 4-14:	Representation of the <i>has-time-interval</i> relation	74
Figure 4-15:	Representation of a state	75
Figure 4-16:	Representation of an aggregate state	75
Figure 4-17:	The representation of a resource	76
Figure 4-18:	The representation of a milling machine	76
Figure 4-19:	The representation of a fixture	76
Figure 4-20:	Representation of a work-order	77
Figure 4-21:	Representation of a part	77
Figure 4-22:	Relations associating a single constraint to a variable	79
Figure 4-23:	Relations linking simple and aggregate variables	79
Figure 4-24:	Representation of a simple variable	80
Figure 4-25:	Relations connecting aggregate variables	80
Figure 4-26:	Representation of an aggregate variable	81
Figure 4-27:	Representation of constraint relaxations	82
Figure 4-28:	Continuous constraint relaxation	82
Figure 4-29:	Discrete constraint relaxation	82
Figure 4-30:	Exclusive and inclusive OR relaxations	83
Figure 4-31:	Representation of the <i>has-relaxation-spec/relaxation-spec-of</i> relations	83
Figure 4-32:	Representation of a constraint	84
Figure 4-33:	Five of Allen's Temporal Relations	85
Figure 4-34:	Representation of the rest of Allen's Temporal Relations	86
Figure 4-35:	Representation of a time object	87
Figure 4-36:	The relations <i>dates</i> and <i>dated-by</i>	88
Figure 4-37:	Representation of a time line	88
Figure 4-38:	Representation of a time interval	89

INTRODUCTION

Many problems that arise in Manufacturing and Strategic Planning involve examining an exponentially growing set of alternatives and finding a solution that satisfies a large number of constraints. While some of these constraints can be expressed in mathematical form, many tend to be heuristic or symbolic in nature requiring the use of powerful Artificial Intelligence-based constraint directed reasoning tools. The theoretical basis for developing these tools is the development of a general theory of constraint-directed reasoning.

Our experience in developing planning and scheduling systems for large problems has demonstrated the importance of being able to reason about constraints. By understanding the constraints and their impact on the structure of the problem space, better results can be generated more efficiently. The theory of constraint-directed reasoning has two parts. The investigation of a theory of constraint-directed reasoning as performed by a single agent, and the investigation of distributed constraint-directed reasoning in a multi-agent problem solving situation. In the case of a single agent, the issues under investigation include:

- The semantics of constraint representation
- How constraints determine the structure of the problem space
- How constraints focus the attention of search
- How constraints diagnose poor decisions and repair them

In the multi-agent case, the general issue is to develop a theory of negotiation using constraints. The issues under investigation include:

- How can the agents coordinate their decisions in order to achieve global behavior
- At what points during problem solving is negotiation effective
- What are the negotiation strategies and protocols

This report summarises the activities and results of the first year's effort on the project that started at the end of November 1987. This effort concentrated on investigating the constraint-directed reasoning behavior of a single agent. The report is divided into three chapters. The first chapter presents the overall problem solving framework, an integration of constraint satisfaction and heuristic search. The second chapter presents the propagation of preferential constraints pertaining to temporal relations and resource capacities. The third chapter describes the focus of attention mechanism of a constraint-directed scheduler and some preliminary experimental results. The fourth chapter presents the representational framework for the concepts used in the research.

CHAPTER 1: Constraint Satisfaction and Heuristic Search

1.1 Introduction

We propose a model of problem solving that provides both structure and focus to search in the problem space. The model achieves this by combining the process of constraint satisfaction (CSP) with heuristic search (HS). The resulting model both reduces search complexity and provides a more formal explanation of the nature and power of heuristics in problem solving. Our model focuses on reasoning within a problem space, and can be viewed as being complementary to the Soar architecture [Laird, Newell & Rosenbloom 87].

Why are problem solving models important? Choosing the right model should lead to the efficient construction of an acceptable quality solution. Two approaches to constructing problem solving models have been taken. The first comes out of the Operations Research tradition where an optimizing algorithm is constructed for a narrow class of problems. For example, the Simplex method finds an optimal solution for problems represented as a set of linear inequalities. The second class of models focuses on general theories of problem solving. For example, at Carnegie Mellon the creation of general models for problem solving dates back to 1956 with the work on Logic Theorist [Newell & Simon 56] to GPS [Newell & Simon 63] to production systems [Newell & Simon 72] and more recently to Soar [Laird, Newell & Rosenbloom 87]. Our interest lies in the latter class of models. In particular, we are concerned with the principles behind how knowledge can be used to structure and guide search in the problem space¹.

We propose a refinement to the heuristic search problem solving model, which we call *Constrained Heuristic Search* (CHS) that retains the synthetic capabilities of heuristics search while obtaining the structural characteristics of constraint-satisfaction techniques². In particular, our model begins with the definition of a problem space composed of states, operators and an evaluation function, and adds two fundamental components:

1. **Problem Space Topology:** Provides a structural characterization of the problem space.
2. **Problem Space Textures:** Provide measures of decision complexity and importance.

¹A problem space is composed of an initial state that defines the problem's initial conditions, a set of operators that generate new states and an evaluation function that identifies solution states.

²Simon [Simon 83] has proposed that there are three "rather distinct ways ... for representing and thinking about problem solving tasks." The first views problem solving as a *search* through a "problem space" of nodes (i.e., states) and links. The second views problem solving as *reasoning*, where new statements are deduced from a set of axioms in a formal language of logic. The third views problem solving as *constraint satisfaction*, where the incremental addition of constraints narrows down a set of objects to a subset which satisfies all the constraints. While these views are not mutually exclusive, they are viewed as being distinct. In fact, a constraint satisfying algorithm is viewed as taking giant steps, not creating new objects, but reducing the entire space of objects to a satisficing set. (This assumes the ability to enumerate a set of objects from which to choose.) On the other hand, search techniques, for example, planning, can be synthetic; incrementally constructing a solution as part of the search process.

1.2 Problem Space Topology

Early notions of sources of efficiency in heuristic search have focused on the concept of the "well structuredness" of the problem space. At least three views of well structured problem spaces exist:

- A problem is not well structured if it cannot be completely defined. This is the case if an objective function cannot be identified or if some problem constraints are not known. Rapid prototyping is a means of eliciting problem structure.
- Simon's [Simon 83] definition of well-structuredness essentially focuses on the ability to operationalize, in a computational sense, the means of solving the problem (i.e. a problem space in order to be well-structured requires a success criterion, a representation structure where states are represented, a set of legal moves between states, etc.)
- Newell [Newell 69] has defined a problem as being "ill structured" if there only exists *weak* methods to solve it. This definition focuses on the performance of the problem solver.

We believe that the third definition is more in line with "conventional wisdom"; it is essentially an issue of problem solving performance, and that problem space structure and focus of attention are important components of problem solving performance.

Within the heuristic search model, a variety of techniques for structuring the problem space have been investigated. ABSTRIPS [Sacerdoti 74] demonstrated how hierarchical reformulation of the problem space via omission of variables reduces search complexity. Hearsay-II [Erman et al 80], MOLGEN [Stefik 81], and OPIS [Smith, Fox & Ow 86] demonstrated how hierarchical reformulation via aggregation/abstraction reduces search complexity. ISIS [Fox 83a] demonstrated how hierarchical reformulation via omission of constraints reduces search complexity. These structuring techniques are more engineering guidelines than formal characterizations.

On the other hand, constraint satisfaction research has begun to formalize the concept of well structured constraint graphs, but their techniques can only be applied to a narrow set of problems. Constraint satisfaction techniques, as described in [Mackworth 77a, Haralick & Elliott 80, Freuder 82a, Dechter & Pearl 87], approach problem solving by constructing a constraint graph where nodes are variables with discrete domains and arcs are n-ary constraints among the values the variables may be assigned. Problem solving is performed by sequentially choosing a variable and a value to assign to it that satisfies all constraints incident upon it. Backtracking occurs when an assignment cannot be found. Research has gone into methods for structuring the network so that the amount of backtracking can be reduced.

CSP methods turn out to be a specializations of heuristic search. In particular, they are restricted to a class of problems we define as *Finite State Problem Spaces* (FSPS).

Definition 1: A Finite State Problem Space has each state defined by a finite set of variables, where each variable's assignment is a subset of a finite, discrete domain.³

³This definition does not include CSP techniques that propagate intervals across variables whose domains are continuous such as in [Smith 83a, Vere 83a].

Solving a problem in a FSPS involves finding an assignment of values to the variables that satisfy a set of constraints. From a problem space perspective, the initial state contains all the variables and their domains, the operators select a variable and a value to assign it, and the evaluation function is composed of the constraints. The sequence of states generated in the search space represent alternative orderings of variables and values to assign to them. Backtracking results in new branches in the search tree. The important insight that we wish to draw from CSP research is that by manipulating the constraint graph, the ordering of variables and values can be optimized. That is, the constraint graph can be viewed as providing a structure for the problem space. Arc-consistency is one such technique that achieves local consistency between groups of variables via the elimination of incompatible values [Montanari 74, Mackworth 77a, Davis 87]. Width 1 networks that are arc consistent are backtrack free.

Based upon the above, we can now provide a description of a problem space's topology. Problem space topology refers to the structure of the problem space that provides a "map" of decision areas. Choosing the area from which to search is the role of the texture measures. Reformulation of a problem results in a different topology. We define problem space topology as a graph where nodes are decision points (i.e. variables or set of variables) and arcs represent the effects that decisions have on each other (i.e. constraints)⁴. Each variable has a (finite or infinite) set of possible values (decisions) associated to it. (This is equivalent to a CSP problem formulation, but in order for constraint graphs to be viable for general heuristic search they have to be extended to include a wider variety of constraints such as preferences, and variables with continuous, non-interval domains.) In the factory scheduling domain a node is often identified as an activity whose possible values are tuples specifying possible resources and times for the activity. Constraints are temporal and causal relations between the activities, and preferences for certain selections.

We distinguish between two types of problem space topologies:

Definition 2: A *completely structured problem space* is one in which all non-redundant nodes (variables) and arcs (constraints) are known a priori.

This is true of all CSP formulations.

Definition 3: A *partially structured problem space* is one in which not all non-redundant nodes and arcs are known prior to problem solving.

This definition tends to be true of problems in which synthesis is performed resulting in new variables and constraints (e.g. the generation of new subgoals during the planning process).

The role of operators within CHS is to create and/or alter topologies, i.e., nodes and constraints. Indeed there are two main types of operators: operators that add structure to the problem space and operators that restructure the problem space.

Features of the problem space topology are the types of nodes and constraints (and their associated propagation algorithms). Davis [Davis 87] mentions, indirectly, a few types of what

⁴Notice that we have assumed binary constraints. In the case of constraints of higher arity, it is convenient to think of the topology as a graph with two types of nodes: variable-nodes and constraint-nodes. Arcs indicate how a constraint affects different variables. The number of arcs coming out of a constraint-node is the arity of the corresponding constraint.

we view as topological features of the problem space, namely the types of values the domain of a node may contain:

- nodes whose domains are discrete and finite (label and value inference)
- nodes whose domains are intervals
- nodes whose domain has a belief for each member (relaxation labelling)
- nodes whose domain are expressions (expression inference)

and types of constraints:

- constraints that are unary predicates,
- constraints that order relations,
- constraints that are bounded differences (e.g. $x - y \geq c$),
- constraints that are linear equations and inequations with unit (i.e. -1, 0, 1) coefficients,
- constraints that are linear equalities and inequalities with arbitrary coefficients,
- constraints that are boolean combinations of constraints,
- constraints that are algebraic equations,
- constraints that are transcendental equations.

Additionally domains may or may not have preferences for values (e.g. preferences for due dates of a job).

The importance of formalizing the concept of problem space topology is that:

- problem spaces have been refined to include the representation of constraints.
- the process of problem reformulation can be formalized as transformations of problem space topological primitives, and
- properties can be proved about the search to be performed, for example, from the CSP literature, width-1 constraint networks that are arc consistent are backtrack free.

1.3 Problem Space Textures

Well focused search is concerned with the ability of the search algorithm to opportunistically decide where in the problem space the next decision is to be made⁵. In order for search to be well focused there must be features of the problem space which differentiate one subspace from another, and these features must be related to the goals of the problem. We have identified and are experimenting with seven such features that we call problem space *textures* [Sadeh & Fox 88]. Below we define these textures for CHSs where all solutions are equally preferred. These definitions can be generalized to the class of Optimization Constraint Satisfaction Problems (OCHSs), where the objective function is a sum of functions of one variable⁶.

⁵The concept of focused search was elaborated in Hearsay-II [Erman 80].

⁶This is done using Bayesian probabilities to approximate the likelihood that a given assignment results in an optimal solution [Sadeh & Fox 88].

- **(Variable) Value Goodness:** the probability that the assignment of that value to the variable leads to an overall solution to the CHS (i.e. to a fully consistent set of assignments). This texture is related to the value ordering heuristics [Haralick & Elliott 80] which look for the least constraining values. Value ordering heuristics are meant to reduce the chance of backtracking. In the case of discrete variables, the goodness of a value is the ratio of complete assignments that are solutions to the CHS and have that value for the variable over the total number of possible assignments.
- **Constraint Tightness:** Constraint tightness refers to the contention between one constraint or a subset of constraints with all the other problem constraints. Consider a CHS A and a subset S of constraints in A. Let B be the CHS obtained by omitting S's constraints in A. The constraint tightness induced by S on A is defined as the probability that a solution to B is not a solution to A. In the case of discrete variables, this is the ratio of solutions to B that are not solutions to A over the total number of solutions to B or equivalently the ratio of solutions to B that do not satisfy S over the total number of solutions to B. Notice that in particular the constraint tightness induced on a redundant constraint is zero. Also the constraint tightness on the set of all constraints of a problem can be used as a measure of the difficulty of the problem. For instance, in the case of discrete variables, this last measure can be expressed as the number of solutions to the CHS over the number of possible complete assignments. Finally notice also that this definition generalizes Nadel's notion of constraint looseness/satisfiability ratio [Nadel 86a] to groups of constraints and to constraints involving both discrete and continuous variables. Moreover Nadel's satisfiability ratio, defined only for a single constraint involving discrete variables, differs from our notion of constraint looseness/tightness in one important way: it only depends on the constraint itself while our definition is with respect to a given CHS. Indeed our notion of constraint tightness also accounts for the other constraints of the problem and their interactions with the constraints whose tightness is being measured.
- **Variable Tightness with respect to a set of constraints:** Again consider a CHS A, a subset S of constraints, and the CHS B obtained by omitting S in A. A variable V's tightness with respect to the set of constraints S is defined as the probability that the value of V in a solution to B does not violate S. In the case of discrete variables, this is simply the ratio of solutions to B in which V's value violates S (i.e. at least one of the constraints in S) over the total number of solutions to B.
- **Constraint Reliance:** This measures the the importance of satisfying a particular constraint. Consider a constraint C. We defined CHS B as being CHS A - {C}. Given that constraints can be disjunctively defined, the reliance of CHS A on a constraint C is the probability that a solutio to CHS B is not a solution to A. In the

case of discrete variables, constraint reliance is defined as the ratio of the number of solutions to CHS B that are not a solution to CHS A over the number of solutions to CHS B. The larger the value, the greater the reliance the problem has on satisfying the particular constraint.

- **Variable Tightness:** Consider a variable V in a CHS A. Let S be the set of constraints involving V (i.e. in a graph with variable-nodes and constraint-nodes, S is the set of constraint-nodes that are directly connected to V) and B be the CHS obtained by omitting S in A. V 's tightness with respect to S is simply called V 's tightness. Hence the tightness of a variable is the probability that an assignment consistent with all the problem constraints that do not involve that variable does not result in a solution. In the case of discrete variables, this is the ratio of the number of complete assignments that are solutions to A over the number of complete assignments that are solutions to B, i.e. the ratio of solutions to the CHS over the number of complete assignments that satisfy all the CHS's constraints except possibly one or more constraints involving V . Alternatively one can define *variable looseness* as the probability that an assignment that has been checked for consistency with all the problem constraints, except those involving that variable, results in a fully consistent assignment. Notice that if one uses a variable instantiation order where V is the last variable, V 's tightness is the *backtracking probability*. Indeed variable looseness/tightness can be identified with variable ordering heuristics [Haralick & Elliott 80, Freuder 82a] which instantiate variables in order of decreasing tightness.
- **Variable Contention:** It estimates the degree of contention that exists among a set of constraints in assigning a value to a variable. Given a CHS A, a set S of constraints incident at variable V , and $\text{CHS } B = \text{CHS } A - S$, one measure of contention is to take the ratio of the number of elements s of the powerset of S that do not have a solution to $\text{CHS } B + s$, to the total number of elements in the powerset of S . In essence, the more combinations of constraints in S for which there is not a solution, the greater the contention.
- **Constraint Arity:** the number of variables involved in a constraint or more generally in a group of constraints.

These textures generalize the notion of constraint satisfiability or looseness defined by [Nadel 86a] and apply to both CHSs (and CSPs) with discrete and continuous variables. We have extended these textures to OCHSs where the objective function is expressed as a sum of functions of one variable, using Bayesian probabilities to approximate the likelihood that a variable results in an optimal value.

Notice that, unless one knows all the CHS's solutions, the textures that we have just defined have to be approximated. Textures may sometime be evaluated analytically [Sadeh & Fox 88]. A brute force method to evaluate any texture measure consists in the use of Monte Carlo

techniques. Such techniques may however be very costly. In general, for a given CHS, some textures are easier to approximate than others, and some are also more useful than others. Usually the texture measures that contain the most information are also the ones that are the most difficult to evaluate. Hence there is a tradeoff. Each domain may have its own approximation for a texture measure.

Textures provide a more formal view of attention focusing. As such, they can explain the power of heuristic knowledge used in search. We have already mentioned variable and value ordering heuristics respectively based on variable looseness and value goodness. Another example is in factory scheduling, where a useful heuristic is to schedule the bottleneck resource first. In our factory scheduling example we show that the concept of resource bottleneck analysis is motivated by constraint arity considerations and illustrates the concept of constraint tightness.

1.4 CHS Problem Solving Process

The CHS model of problem solving is a combination of constraint satisfaction and heuristic search. In particular we view a problem space's topology as a mesh that envelopes the problem space. Propagation of constraints results in the tightening of the mesh around and through the problem space, thus reducing the number of alternatives to be considered and/or generated. In completely structured topologies, the nodes represent islands at which decisions are to be made. For partially structured topologies, nodes represent starting points at which operators are to generate new nodes.

The problem solving model we propose contains the following elements:

- search begins with a single state in which the initial problem space topology (i.e. constraint mesh) is constructed.
- constraint propagation through the mesh is performed
- texture measures are computed
- decision nodes are identified
- a new state is generated by selecting an operator that either adds structure to the topology or further restricts the domain of a variable

The next two sections demonstrate the application of the CHS model to the problems of spatial planning and factory scheduling.

CHAPTER 2: Propagating Temporal and Capacity Preferences

2.1 Introduction

2.1.1 The Issue

We are concerned with the issue of how to opportunistically focus an incremental scheduler's attention on the most critical decision points and the most promising decisions in order to reduce search and improve the quality of the resulting schedule. More specifically we are concerned with incremental constraint directed scheduling where the problem is defined as a set of variables and a set of constraints. Both variables and constraints are determined by the initial scheduling problem and the earlier decisions made by the scheduler. The interactions of the constraints determine the structure of the problem space. We characterize the problem space with a set of texture measures that are used to both identify critical decision points and select a decision at each of these points. The process of analyzing the current problem and generating new decisions (e.g. scheduling an operation) is repeated, thereby resulting in the incremental construction of a schedule.

Real-life scheduling problems are subject to a variety of preferences [Johnson 74, Fox 83b, Ow 84, Smith 86] such as meeting due dates, reducing the number of machine set-ups, reducing inventory costs, using accurate and/or fast machines, making sure that some jobs are performed within a single work-shift, etc. Although these preferences are usually set independently to one another, they interact. For instance selection of a good start time for an activity (e.g. to meet a due date) may prevent the selection of an accurate machine for another operation or may prevent meeting another job's due date. For this reason, selecting operation start times or allocating resources based solely on local a priori preferences is likely to result in poor schedules. Preference propagation is meant to allow for the construction of measures that reflect preference interactions. These measures can then serve to guide the construction of a good *overall* schedule rather than a schedule that locally optimizes a subset of preferences.

We perform preference propagation within a probabilistic framework. We associate with each variable's value a probability that reflects the likelihood that this value results in a good schedule overall (*value goodness*). These probabilities are refined by being propagated across the problem constraints. Value goodness is a texture measure that helps selecting assignments for variables. Identification of critical variables (i.e. decision points) is performed using another texture measure, called *variable looseness*. A critical variable or group of variables is one whose good overall values are likely to become unavailable if one were to start assigning values to other variables first. Notice that if our measures of value goodness were perfect, the order in which variables are instantiated would not matter. However such perfect measures could only be obtained by first solving the problem. Because in practice measures of value goodness contain some uncertainty, one has to account for the effects of assigning a value to a variable over the availability of good values for other variables. A variable instantiation order is accordingly defined starting with the most critical (i.e. least loose) variables. In this paper we are interested in the identification of critical activities (i.e. operations). An activity is made of a start time variable, possibly a duration variable, and a set of resource variables. We identify critical

activities as the ones that heavily rely on the possession of highly contended resources. Indeed, if such critical activities are not scheduled first, it is very likely, that by the time the scheduler turns its attention to these activities, the resources that would have been the most appropriate for these activities will no longer be available.

We discuss preference propagation in temporal/capacity constraint graphs (T/CCG). Temporal constraints define partial orderings among the activities to be scheduled. All thirteen of Allen's [Allen 84] temporal relation constraints are accounted for. Resource capacity constraints restrict the use of resources to only one activity at a time. Both situations with fixed and variable duration activities are discussed. Our formalism allows for both activity start time and duration preferences as well as for resource preferences. It also accounts for prior resource reservations if any. It is shown that the (a posteriori) start time and duration distributions resulting from the propagation across the temporal constraints can be combined to identify resources that are highly contended for (*resource contention*) and activities that heavily rely on the possession of these resources (*activity resource reliance*).

We also argue that a posteriori start time/duration distributions can be seen locally as measures of start time/duration goodness and globally as measures of start time/duration looseness. Our notion start time/duration looseness generalizes the Operations Research notion of slack [Baker 74, Johnson 74].

2.1.2 Formalization of the Scheduling Problem

The factory scheduling problem is often described as a two step problem: a *process planning* step and a *resource planning* step [Fox 83b]. Process planning deals with the generation and selection of plans (i.e. process routings) that satisfy the order specifications. Resource planning, sometimes also referred to as scheduling, deals with the allocation of resources (e.g. machines) to activities and the assignment of start and end times to activities. In general both steps can be interleaved.

In this paper we will be concerned exclusively with the scheduling part of the problem: we will assume that we are given a set of plans to schedule. Here a plan is simply defined as a partial ordering of activities. Each activity may require one or more resources, for each of which there can be several alternatives.

We formalize the scheduling problem as a constraint satisfaction problem (CSP). The variables of the problem are the activity start times, the resources allocated to each activity, when there is a choice, and possibly the duration of each activity. An activity's end time is defined as the sum of the activity's start time and duration. We differentiate between two types of constraints: *required constraints* and *preferential constraints* [Fox 83b]. Required constraints determine the admissibility of a solution to the CSP (schedule) while preferential constraints allow for differentiating among admissible solutions. The degree of satisfaction of a preferential constraint is defined by a utility function that maps the possible values of a variable onto utilities ranging between 0 and 1. A utility of 0 indicates a non-admissible value. A value with utility 1 is an optimal value.

We will be dealing explicitly with two types of required constraints: *temporal relation constraints* and *resource capacity constraints*. Temporal relation constraints are used to describe

Relation -----	Pictorial Representation -----
X BEFORE Y Y AFTER X	XXX YYY
X EQUALS Y Y EQUALS X	XXX YYY
X MEETS Y Y MET-BY X	XXXYYY
X OVERLAPS Y Y OVERLAPPED-BY X	XXX YYY
X DURING Y Y CONTAINS X	XXX YYYYY
X STARTS Y Y STARTED-BY X	XXX YYYYY
X FINISHES Y Y FINISHED-BY X	XXX YYYYY

Figure 2-1: Allen's 13 temporal relation constraints

partial orderings among activities as provided by the process planning step. We will be using Allen's temporal relation constraints [Allen 84] to describe these constraints (Figure 2-1). We will refer to the graph defined by these constraints, for a given CSP, as the CSP's *temporal constraint graph* (TCG). Capacity constraints restrict the number of reservations of a resource over any time interval to the capacity of that resource. In this paper, for the sake of simplicity, we will always be assuming resources with unary capacity. Together these required constraints form a temporal/capacity constraint graph (T/CCG). A schedule that does not satisfy the required constraints of the CSP is not admissible.

We will allow for preferential constraints on activity start times and durations as well as on the resources to be used by each activity. Preferential constraints are described with utility functions. For a given preferential constraint, a variable's value is admissible only if its utility is strictly positive. High preference for an admissible value is indicated by a high utility. In practice the domain of admissible values resulting from these preferential constraints, i.e. the domain with strictly positive utilities, is always bounded. For instance the domain of admissible start times of an activity is constrained at one end by the order release date and at the other end by the order due date according to the durations of the activities that precede/follow the activity within the plan.

Notations

We have to schedule a set of activities $\{A_1, A_2, \dots, A_n\}$. Let I_k denote the time interval over which A_k spans. st_k , et_k and du_k respectively denote I_k 's start time, end time, and duration.

Activities are connected by a set of temporal relation constraints, thereby forming a TCG. We view TCGs as *undirected* graphs. An arc in a TCG indicates the presence of a temporal relation between two intervals (e.g. I_1 BEFORE I_2 or equivalently I_2 AFTER I_1). Let C_1, C_2, \dots, C_m denote the temporal relation constraints in the TCG. The TCG, which has been produced during the process planning phase, is assumed consistent.

Additionally there are capacity constraints limiting the use of each resource to only one activity at a time. By adding these capacity constraints to the TCG, one obtains the CSP's T/CCG. In this paper we will not need to formalize the description of T/CCGs any further as we will be mainly dealing with their temporal abstractions, i.e. the TCGs obtained by omitting the capacity constraints in the T/CCGs.

Each activity A_k has a preferential start time constraint with associated utility function denoted u_{st_k} . Activity A_k 's duration is either fixed or constrained by a preferential constraint with utility function u_{du_k} . The ranges of admissible start times and durations are assumed to be bounded, which is always the case in practice.

Each activity A_k may require one or more resources $R_{k1}, R_{k2}, \dots, R_{kp}$. For each resource R_{ki} required by activity A_k , there is a set of possible resources $R_{ki1}, R_{ki2}, \dots, R_{kij}$ available on the factory floor. This set is assumed to be finite, which is also always the case in practice. These different resources are usually not equally preferred. A resource utility function, $u_{R_{ki}}$ associates a utility (preference) $u_{R_{ki}}(R_{kij})$ with each possible resource R_{kij} . For instance a milling operation may require a milling machine and a human operator. There may be two milling machines available on the factory floor. For this specific milling operation, milling-machine₁ may have a preference of 1.0 and milling machine₂ a preference of 0.4 (e.g. due to a difference in the accuracy of the machines.). There may also be several human operators available with different utilities.

The global utility of a schedule is obtained by summing all the preferential constraints' utilities (for the given schedule).

2.1.3 An Example

We now introduce a simple scheduling problem that we will use throughout this paper.

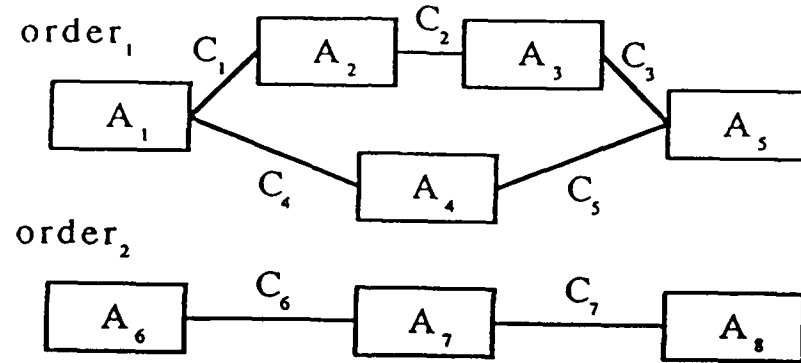
The problem involves scheduling two orders: order₁ and order₂ (Figure 2-2):

- order₁ comprises five activities: A_1, A_2, \dots, A_5 ,
- order₂ comprises three activities: A_6, A_7 , and A_8 .

All activities have the same duration, namely 30 time units. C_1, C_2, \dots, C_7 are the temporal relation constraints imposed by the process planning step. For instance, C_1 indicates that A_1 has to precede A_2 . The domain comprises three physical resources: R_1, R_2 , and R_3 .

- A_1 requires a resource R_{11} which can be either R_1 or R_2 (with equal preference), i.e. $u_{R_{11}}(R_1)=u_{R_{11}}(R_2)=1$, and $u_{R_{11}}(R_3)=0$.
- A_2 requires a resource R_{21} which has to be R_1 , i.e. $u_{R_{21}}(R_1)=1$, and $u_{R_{21}}(R_2)=u_{R_{21}}(R_3)=0$.

- A_3 requires a resource R_{31} which has to be R_2 , i.e. $u_{R_{31}}(R_2)=1$, and $u_{R_{31}}(R_1)=u_{R_{31}}(R_3)=0$.
- A_4 requires a resource R_{41} which has to be R_2 , i.e. $u_{R_{41}}(R_2)=1$, and $u_{R_{41}}(R_1)=u_{R_{41}}(R_3)=0$.
- A_5 requires a resource R_{51} which has to be R_1 , i.e. $u_{R_{51}}(R_1)=1$, and $u_{R_{51}}(R_2)=u_{R_{51}}(R_3)=0$.
- A_6 requires a resource R_{61} which has to be R_3 , i.e. $u_{R_{61}}(R_3)=1$, and $u_{R_{61}}(R_1)=u_{R_{61}}(R_2)=0$.
- A_7 requires a resource R_{71} which can be either R_2 or R_3 (with equal preference), i.e. $u_{R_{71}}(R_2)=u_{R_{71}}(R_3)=1$, and $u_{R_{71}}(R_1)=0$.
- A_8 requires a resource R_{81} which has to be R_3 , i.e. $u_{R_{81}}(R_3)=1$, and $u_{R_{81}}(R_1)=u_{R_{81}}(R_2)=0$.



- C_1 : A_1 BEFORE A_2
 C_2 : A_2 BEFORE A_3
 C_3 : A_3 BEFORE A_5
 C_4 : A_1 BEFORE A_4
 C_5 : A_4 BEFORE A_5
 C_6 : A_6 BEFORE A_7
 C_7 : A_7 BEFORE A_8

Figure 2-2: TCG for a two order scheduling problem

Activities in $order_1$ (i.e. A_1, \dots, A_5) are assumed to have the same start time utility function. The function requires that these activities start between time 0 and time 140, with an optimal start time at 120 (Figure 2-3). Time 0 can be interpreted as the order release date. Time $140 + 30 = 170$ (latest-start-time + duration = latest-finish-time) could represent the time after which the client would refuse the order.

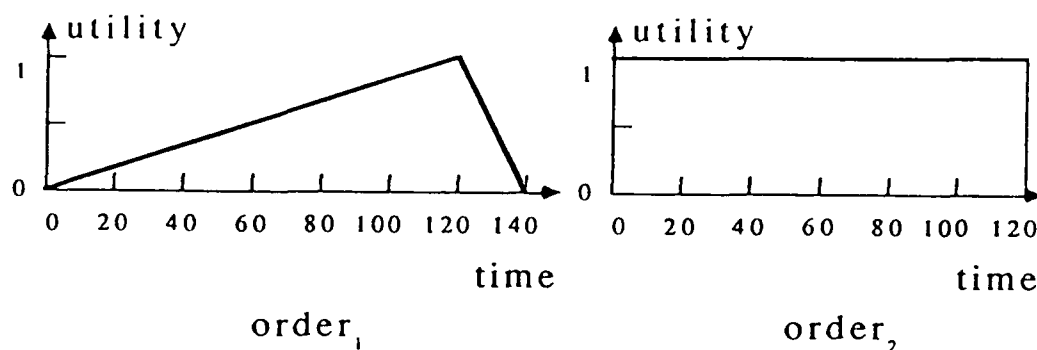


Figure 2-3: Start time utility functions

All activities in $order_2$ will be assumed to have a uniform start time utility between 0 and 120 (Figure 2-3).

In general preferential constraints are set independently to one another and may therefore be incompatible. For instance, it will obviously be impossible to simultaneously schedule both A_2 and A_3 at their optimal start time, namely 120. Therefore instead of a priori preferences, one needs (a posteriori) preferences indicating values that are likely to result in a good schedule *overall*. Such preferences can only be obtained by accounting for constraint interactions. This is the objective of the propagation technique presented in this paper.

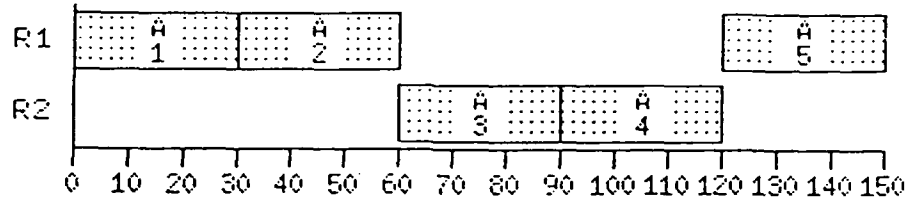
Looking more closely at our example, one notices that there are four activities requiring resource R_2 : A_1 , A_3 , A_4 , and A_7 . Out of these four activities, up to three can occur in parallel, namely A_3 , A_4 , and A_7 . These activities will therefore compete for the possession of R_2 . There is no such competition for R_1 and R_3 as the activities that require these resources are fully ordered temporally (e.g. A_1 has to be carried out before A_2). The scheduling of R_2 is therefore more critical than that of R_1 and R_3 . An incremental scheduler should first focus its attention on the competition between A_3 , A_4 , and A_7 for R_2 . Moreover, since A_7 has two resource alternatives (R_2 and R_3) while A_3 and A_4 have only one, and since A_3 has less slack than A_4 , we would like our incremental scheduler to first schedule A_3 with R_2 ⁷.

A more detailed analysis confirms that first scheduling A_3 rather than A_4 is the right decision. It also reveals the influence of the preferential constraints (in this case the start time preferences) in determining activity criticality. We consider two scenarios: one where the first activity to be scheduled is A_4 (scenario1) and one where it is A_3 (scenario2). Given that A_5 cannot start later than 140 and that A_3 and A_4 have a duration of 30, A_1 , A_2 , A_3 , and A_4 cannot start later than 110. Hence, according to their start time utility functions, these activities will prefer to start as late as possible (Figure 2-3). In scenario1, A_4 is the first activity to be scheduled. It is scheduled as late as possible (while still leaving some room for A_5 to have a good schedule), say at time 90. Since both A_3 and A_4 require R_2 , A_3 has to be scheduled before A_4 . The resulting schedule is the

⁷Notice that we are assuming an incremental scheduler whose reservations are nonpreemptible. The order in which activities are allocated resources would not matter if allocations were preemptible. Most predictive schedulers do not allow for such preemptions as they tend to produce infinite loops if one does not take special precaution.

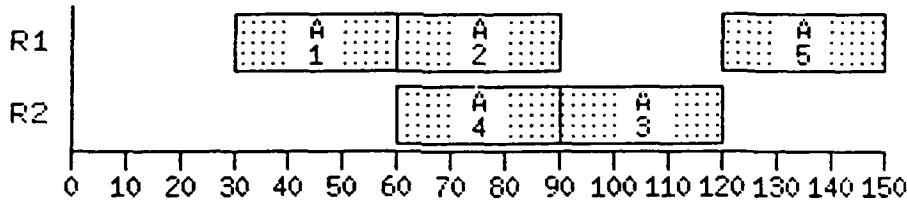
sequence A_1, A_2, A_3, A_4 , and A_5 , as displayed in Figure 2-4. Alternatively (scenario2) suppose that we decide to first schedule A_3 . This time A_3 is scheduled to start at time 90, and A_4 has to occur before A_3 . However A_2 and A_4 can occur in parallel before A_3 (Figure 2-5). Hence, globally, A_1 to A_4 start later in scenario2 than in scenario1. In other words, scenario2 results in a higher global utility than scenario1.

Our approach to preference propagation formalizes the above considerations.



$$u_{st_1}(0) + u_{st_2}(30) + u_{st_3}(60) + u_{st_4}(90) + u_{st_5}(120) = 2.5$$

Figure 2-4: Gantt chart for a schedule of order1 obtained with scenario1.
The global start time utility has been obtained by adding the start time utility of the five activities.



$$u_{st_1}(30) + u_{st_2}(60) + u_{st_3}(90) + u_{st_4}(60) + u_{st_5}(120) = 3.0$$

Figure 2-5: Gantt chart for a schedule of order1 obtained with scenario2.
The global start time utility has been obtained by adding the start time utility of the five activities.

2.1.4 Organization of the Paper

In the next section we give an overview of related work in constraint satisfaction and scheduling. Section 3 introduces the assumptions that are the basis to our probabilistic approach to preference propagation and gives an overview of the approach. Section 4 describes the propagation of start time and duration probability distributions in TCGs. As already mentioned

earlier, when one is simply concerned with knowing whether two time intervals are temporally related or not, a TCG can be considered as an undirected graph. *When we will be talking about cycles in the TCG, we will always be referring to the undirected interpretation of the graph.* Subsection 4.2 deals with acyclic TCGs with fixed-duration activities. Subsection 4.3 relaxes the fixed-duration hypothesis. Subsection 4.4 relaxes the acyclicity assumption. Subsection 4.5 discusses propagation in general TCGs where there are explicit disjunctions of temporal relation constraints. Finally subsection 4.6 analyzes the results of the section with respect to a set of requirements and desiderata identified in subsection 3.2. Section 5 explains how the results of the previous section can be combined to obtain measures of resource contention and activity resource reliance. In section 6, we discuss the time complexity and the expressiveness of our framework as well as possible improvements. Section 7 summarizes the main ideas of the paper.

For the sake of concision, subsections 4.2 and 4.3 contain only the treatment of two temporal relation constraints (BEFORE, and MEETS). Formulas for the complete set of Allen's temporal relation constraints is presented in appendix 1. For the same reason, section 5 only sketches the computation of resource demand densities. The reader will find a complete treatment of these densities in appendix 2.

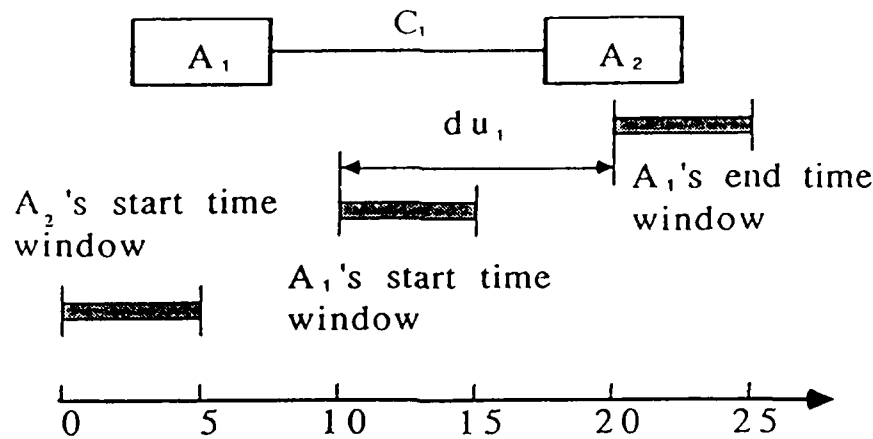
2.2 Related Work

We already mentioned that this paper does not deal with process planning. Hence it is assumed that the TCGs to be scheduled are consistent. [Vilain 86] has proved that consistency checking in a general TCG is NP-hard. Several algorithms have been proposed in the literature to perform partial or total consistency checking in a TCG. Allen's algorithm [Allen 83] achieves 3-consistency⁸ in a general TCG in polynomial time and space. A complete consistency checking algorithm using a variation of data dependency backtracking is presented in [Valdes-Perez 87]. Although the algorithm is designed for quick pruning, its asymptotic complexity remains exponential. [Vilain 86] and [Tsang 87] point out that consistency checking can actually be performed in polynomial time provided that the TCG does not contain certain types of disjunctive relations such as "Interval₁ has to be *either* BEFORE *or* AFTER Interval₂".

When additional constraints such as capacity constraints, preferential start time constraints, preferential duration constraints, preferential resource constraints, or resource reservations are added to a consistent TCG, the resulting CSP may stop being consistent. These are the types of inconsistencies that we will be referring to later in this paper. Consider the simple example depicted in Figure 2-6. There are two activities: A₁ and A₂. A₁ is BEFORE A₂. A₁ has a preferential start time constraint specifying that it has to start between 10 and 15. A₂'s preferential start time constraint specifies that A₂ has to start between 0 and 5. A₁'s duration is 10. A₂'s duration does not matter. The resulting CSP is obviously inconsistent (unsatisfiable) as A₁'s earliest end time (10+10=20) is after A₂'s latest start time (5).

Propagation of activity start and end time windows (earliest/latest start and end times) dates back to the CPM algorithm [Johnson 74]. The PERT method generalizes CPM by allowing for

⁸According to [Freuder 82b], a constraint graph is *k-consistent* if for any set of (k-1) variables, any consistent assignment of values to these (k-1) variables, and any k-th variable, there always exists a value for the k-th variable such that the k values taken together (i.e. for the (k-1)+1 variables) are consistent.



$C_1: A_1 \text{ BEFORE } A_2$

$du_1: A_1 \text{'s duration}$

Figure 2-6: An inconsistent CSP with a consistent TCG

uncertainty in activity durations. [Vere 83b] adapted the CPM propagation techniques to a planning system called DEVISER. In DEVISER start time windows are described as triples of the form (*earliest-start-time, ideal-start-time, latest-start-time*). The ideal start time information is optional. A start time triple can be seen as a triangle-shaped⁹ start time utility function (Figure 2-7). When the ideal start time is omitted, the window can be interpreted as a rectangle-shaped start time utility function. In DEVISER start time windows are dynamically compressed to account for new temporal relations and activities introduced during the planning process. The compression does not account for ideal start times which remain fixed. DEVISER is able to handle both BEFORE/AFTER and MEETS/MET-BY relations. Because preferences are not propagated, the system performs poorly when it has to select a start time within a compressed window: the selection is based on purely local information.

A variation of Vere's algorithm is presented in [Bell 84] that accounts for variable duration activities. [Smith 83b] extends Vere's approach by also accounting for DURING/CONTAINS temporal relation constraints (see also [LePape 87]). Smith's temporal module also allows for the propagation of resource reservations over T/CCGs. [Rit 86] describes a Waltz algorithm to propagate generalized temporal windows over TCGs. A generalized window is the composition of a start time, end time, and duration window (Figure 2-8). The method accounts for all 13 of Allen's temporal relations as well as for disjunctions among these relations. In the general case, Rit's algorithm is only guaranteed to achieve arc-consistency [Mackworth 77b]. Total consistency can however be guaranteed in TCGs that do not contain disjunctions of temporal relation constraints.

None of the propagation techniques that we just described handles preferences. In practice, however, different times within a window are not equally preferred. For instance, in the factory scheduling domain, due dates and associated late delivery penalties induce preferences on

⁹This is not the only possible interpretation.

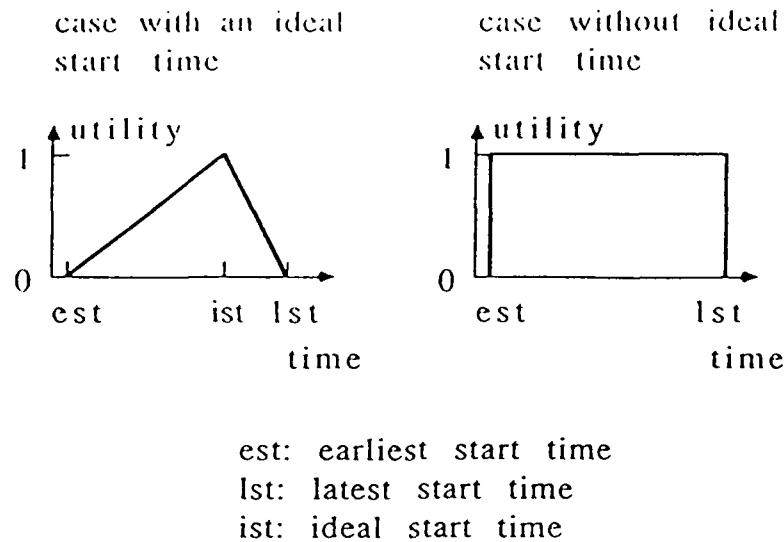


Figure 2-7: A utility interpretation of DEVISER's start time windows

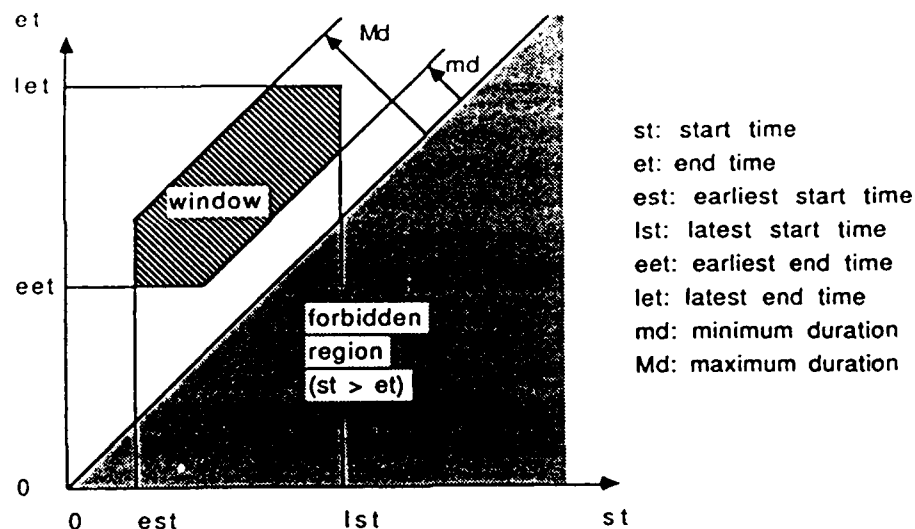


Figure 2-8: Rit's generalized window

activity end times (and hence start times). Inventory costs are another source of start time preferences. In general preferences cannot be accounted for independently. Selection of the optimal start time for one activity may prevent selection of the optimal start time for another either because of temporal relation constraints between the two activities or because of capacity constraints, or a combination of the two. This is why it is crucial not only to propagate time windows but also to propagate preferences over these windows. Window propagation simply guarantees admissibility of the values within a compressed window. *Preference propagation strives not only for admissibility but also for optimality by locally reflecting preference interactions.*

Our purpose is to develop preference propagation techniques to guide an incremental scheduler. Both empirical and analytical studies reported in [Haralick 80, Freuder 82b, Purdom

83, Nadel 86b, Nadel 86c, Nadel 86d, Stone 86] indicate that, in general, the amount of search required to find a solution to a CSP can be significantly reduced by using the following two look-ahead schemes [Dechter 88]:

1. Variable Ordering: Focus on the most constrained variables first.

2. Value Ordering: Try the least constraining values first.

Tightly constrained variables and constraining values are determined by the interactions of the problem constraints. In CSPs where variables have finite sets of possible values and all values are equally preferred, the number of possible values left after constraint propagation (i.e. consistency checking) can be used to determine variable tightness/looseness. In problems where values are not equally preferred, like in the scheduling domain, this is not sufficient. One has also to account for *value goodness*, i.e. the utility of a value and the impact of selecting that value on the availability of good values for the other variables. *Our notion of value goodness and our generalization of the notion of variable looseness are intended to allow for the generalization of these two look-ahead strategies to CSPs where variables can have infinite bounded sets of possible values with non-uniform preferences.*

[Muscettola 87] presents a probabilistic framework to compute resource contention in T/CCGs with only BEFORE/AFTER temporal relation constraints based on assumptions on the order in which the activities are scheduled. This paper extends Muscettola's approach for computing resource contention by removing the need for assumptions on the order in which activities are scheduled, by dealing with all of Allen's constraints, by allowing for duration and resource preferences and by accounting explicitly for earlier resource reservations.

2.3 A Probabilistic Framework for Preference Propagation

2.3.1 An Overview of the Approach

Our purpose is to develop preference propagation techniques to guide an incremental scheduler. An incremental scheduler works by iterating through a two-phase process. In the first phase it analyzes the structure of the CSP resulting from the initial scheduling problem and the decisions that have already been made. In the second phase, based on this analysis, new decisions are generated resulting in the expansion of the current schedule (e.g. new activities are scheduled). If the scheduler reaches a deadend, it backtracks. The process goes on until a satisfactory schedule is produced.

The first phase analysis is performed using preference propagation to dynamically identify critical decision points. Such decision points are determined by the interactions of the problem constraints. In the case of the scheduling problem, there are two main types of interactions: operation precedence interactions and resource requirement interactions [Smith 85].

1. The operation precedence interactions are the ones induced by the TCG. They are sometimes also referred to as *intra-order* interactions¹⁰.
2. Resource requirement interactions are induced by the capacity constraints. They arise from the contention of several activities for the same resource. They are

¹⁰Although activities within a same order can also interact by competing for the same resources.

sometimes referred to as *inter-order* interactions.

Intra-order and inter-order interactions have respectively motivated so-called order-based and resource-based scheduling techniques. In the past few years it has become clear that efficient scheduling requires the ability to combine these two perspectives [Smith 85, Smith 86] so as to account for both types of interactions. Unfortunately both types of interactions are not totally independent. Intra-order interactions affect the time intervals over which activities will contend for resources, thereby influencing inter-order interactions. Resource contention in turn restricts the times over which activities can occur, thereby influencing intra-order interactions.

In order to deal with the uncertainty in the interactions between uninstantiated variables, we have adopted a probabilistic model. For each uninstantiated variable, a probability density is computed for the variable's possible values that indicates the likelihood of each value to result in a good schedule overall, given the decisions already made by the incremental scheduler. This probability is a dynamic measure of value goodness. When the corresponding probability density is normalized, we also interpret this probability as the dynamic probability that the scheduler assigns that value to the variable.

In its simplest form our approach involves the following steps:

1. Based on the current partial schedule as well as start time, duration and resource preferences, *a priori* probability distributions are produced for the start time, duration and resources of each unscheduled activity,
2. These *a priori* probability distributions are propagated over the TCG, resulting in *a posteriori* start time and duration probability distributions,
3. The *a posteriori* distributions obtained in the previous step are combined to compute activity individual demand densities. An activity A_k 's *individual demand density* at time t for a resource R_{kij} , say $D_{kij}(t)$, is defined as the probability that A_k is active at time t and uses R_{kij} to fulfill its resource requirement R_{kij} ,
4. Activity individual demand densities are combined to measure resource *aggregate demand densities*. The aggregate demand density for a resource at time t is the probabilistic demand for that resource at time t .

Both iterative and hierarchical variations of this basic propagation algorithm will be discussed.

From an Operations Research point of view, activity *a posteriori* start time and duration distributions provide a measure of intra-order interactions and a generalization of the notion of *slack* [Johnson 74]. Resource aggregate demand densities reflect the level of resource contention defined by the CSP's inter-order interactions. Resource aggregate demand densities can be identified with the Operations Research concept of *bottleneck analysis* [Smith 85, Smith 86, Muscettola 87].

From a constraint satisfaction perspective, *a posteriori* start time/duration distributions should be regarded locally as measures of *value goodness*, and globally as measures of *variable looseness* for activity start times and durations. Aggregate demand densities can be interpreted as measures of *constraint contention* and individual demand densities as measures of *activity resource reliance*.

2.3.2 Building A Priori Probability Distributions Based on Local A Priori Preferences

Our approach uses Bayesian probabilities to estimate value goodness, i.e. the likelihood that a given value will result in a good schedule overall. It consists in the construction of a priori probability distributions for each uninstantiated variable based on local preferences. These probabilities are then refined so as to account for constraint interactions, thereby resulting into a posteriori probability distributions.

Obviously the main concern in such an approach is to obtain good estimates of value goodness (**desideratum1**) as these estimates are essential to both the identification of critical decision points in the search space and the selection of a decision at these points. There are however some more specific requirements and desiderata to which one should give special attention. Indeed, besides its need for good focus of attention mechanisms, efficient search also requires the ability to quickly prune deadend paths in the search tree.

In our probabilistic framework, unsatisfiability is detected when a posteriori probability densities are uniformly zero. This indicates that the interactions of the problem constraints have reduced the set of admissible values for a variable to the empty set. Detecting unsatisfiability in this fashion requires that:

1. **Requirement1**: Every value that is a priori admissible¹¹ is given a strictly positive a priori probability, though possibly very small.
2. **Requirement2**: The propagation step, which combines a priori probabilities to account for constraint interactions, produces a posteriori probability densities that are zero only for values forbidden by the constraint interactions.

Requirement1 restricts the construction of a priori probability distributions. **Requirement2** is a restriction on the propagation method itself.

Desideratum2: Additionally, in order to detect and prune inconsistent states as soon as possible, one would like a posteriori probabilities to be zero for all non-admissible values (complete consistency checking). Unfortunately interactions between the resource requirements of unscheduled activities seem computationally very expensive to totally account for. Consequently we will have to settle for partial consistency checking.

These general requirements and desiderata having been identified, we turn our attention to the construction of a priori probability distributions. We start with some general observations.

In the presence of a unique variable with a single (unary) preferential constraint, one can just select one of the optimal values defined by the utility function. The probability density for the variable's value consists of a set of peak distributions (Dirac distributions), each centered around one of the optimal values (Figure 2-9b).

On the other hand, in the presence of several variables and constraints, it is not always possible any more to simultaneously select an optimal value for each variable. For instance, it is not

¹¹Of course, a more sophisticated analysis will result in the rejection of a larger number of possible values (see **desideratum2**). Therefore what is really important for **requirement1** is that *no* value received a zero a priori probability while it could have resulted in an admissible schedule.

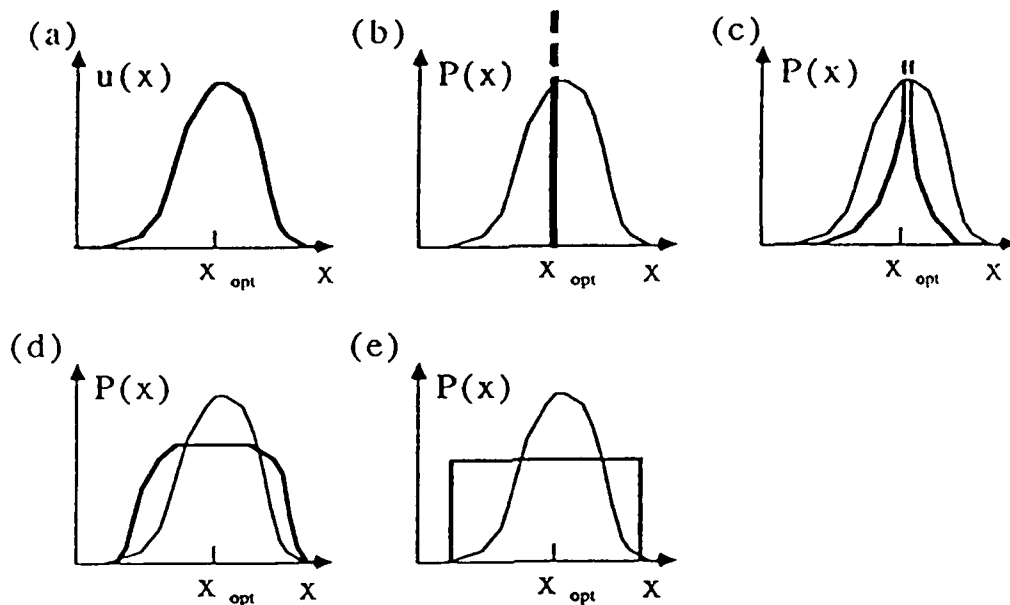


Figure 2-9: A priori probability density $P(x)$ for a variable x with utility function $u(x)$.
This is an example with a single optimal value.

always possible to schedule an activity at its optimal start time and with a set of optimal resources. Very often one has to settle for suboptimal start times and/or resources in order to find a feasible schedule (i.e. satisfy all the CSP's constraints). If, for a given variable, the interactions defined by the constraints are weak, it is usually possible to select a value that is still very close to the optimum (or one of the optima). As interactions become stronger, it becomes more difficult to select values close to the optima: the probability density widens (Figure 2-9c and d). In situations of extremely strong interactions, one is just happy to find a solution within the domain of admissibility (non-zero utility value). Hence the probability distribution tends towards a uniform distribution over the range of admissible values (Figure 2-9e).

The a priori probability densities that we are currently using are essentially obtained by normalizing utility functions. Intuitively this can be interpreted as a sort of average difficulty assumption (see Figure 2-9). The resulting a priori probabilities obviously satisfy requirement 1. Moreover, because we assume that domains of admissibility defined by utility functions are bounded, normalization is always possible.

Prior to normalizing a utility function, its domain is pruned to account for earlier resource reservations. This improves the quality of the probability distributions with respect to desideratum 2. In particular we remove from the start time probability distributions the start times that are not allowed by the current resource reservations. A start time t is not allowed for an activity A_k , if there is at least one resource R_{ki} required by A_k such that none of the resources R_{kij} is totally available between t and $t + \delta_{k_{min}}$, where $\delta_{k_{min}}$ is A_k 's smallest admissible duration.

We are currently investigating alternative methods for producing a priori probability

distributions. In particular we are investigating both iterative and hierarchical approaches to preference propagation. In an iterative approach one can use the resource demand densities obtained by the previous iteration to estimate the probability that a given resource will be available for an activity at some time t . Using these probabilities, new a priori start time probability distributions can be obtained and the propagation process can be carried out all over again. Alternatively, in a hierarchical scheme, one can use the propagation results obtained at an upper level to compute resource availability estimates. Again these estimates can be combined with the start time utility functions to obtain a priori start time probability distributions for the new level.

2.4 Propagating Start Time and Duration Distributions in a TCG

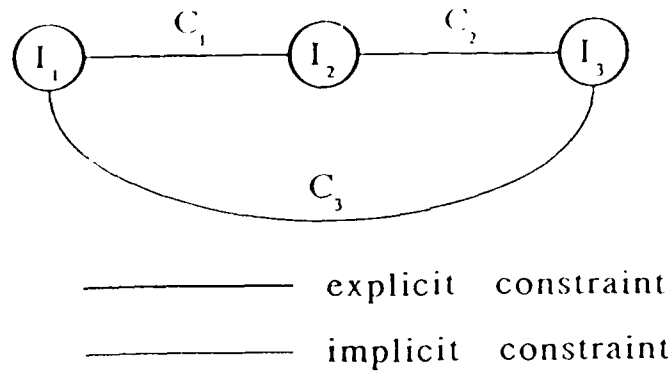
2.4.1 Preliminary Remarks

Now that we have some a priori distributions for the start time, duration (if variable) and resources of an activity, we can refine these a priori probabilities so as to account for the actual constraints of the problem. In this section we compute a posteriori start time (and duration) probabilities that account for the interactions defined by the TCG.

This section is subdivided into several subsections each dealing with the propagation problem under increasingly more general assumptions. As we already mentioned earlier we view TCGs as undirected graphs. Subsection 4.2 develops the computation of a posteriori start time probability distributions in acyclic TCGs with fixed duration activities. In subsection 4.3 the fixed-duration assumption is relaxed. In subsection 4.4 we relax the acyclicity assumption. Subsections 4.2 to 4.4 all assume that there are no explicit disjunctions in the TCG. The transitivity properties [Allen 83] of the relations may however induce disjunctions, which are then implicitly accounted for in the propagation. An example of TCG with no explicit disjunction is represented in Figure 2-10. The TCG specifies that I_1 OVERLAPS I_2 , and I_2 is DURING I_3 . This implicitly induces the disjunction I_1 {DURING, STARTS, OVERLAPS} I_3 , i.e. I_1 is DURING or STARTS or OVERLAPS I_3 . All our calculations allow for this type of implicit disjunctions. On the other hand explicit disjunctions are more difficult to handle and are quite infrequent in practical factory scheduling problems. Propagation in TCGs with explicit disjunctions is discussed in subsection 4.5. Figure 2-11 displays a TCG with explicit disjunctions. Subsection 4.6 interprets the results.

Notations

- C_1, C_2, \dots, C_m will denote the explicit temporal relation constraints that define the TCG (See Figure 2-10 for an example).
- $\sigma_k(st_k=t)$ will denote I_k 's a priori start time probability density, obtained as suggested in the previous section. t is the variable.
- $\delta_k(du_k=d)$ will denote I_k 's a priori duration probability density, obtained as suggested in the previous section. d is the variable. This distribution will only be used for variable-duration activities.
- $P(st_k=t \& C_1 \& C_2 \& \dots \& C_m)$ (where A_k is assumed to be a fixed-duration activity)

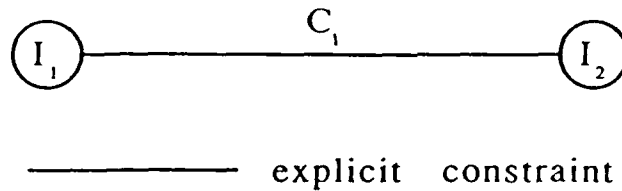


$C_1: I_1 \text{ OVERLAPS } I_2$

$C_2: I_2 \text{ DURING } I_3$

$C_3: I_1 \{ \text{DURING, STARTS, OVERLAPS} \} I_3$

Figure 2-10: Example of a TCG with no explicit disjunctions



$C_1: I_1 \{ \text{BEFORE, AFTER} \} I_2$

Figure 2-11: Example of a TCG with explicit disjunctions

will denote I_k 's a posteriori start time probability density. This density (with variable t) corresponds to the a posteriori probability that A_k starts at time t (i.e. $st_k=t$) and that the temporal relation constraints C_1, C_2, \dots, C_m are satisfied.

- $P(st_k=t \& du_k=d \& C_1 \& C_2 \& \dots \& C_m)$ (where A_k is assumed to be a variable-duration activity) will denote the two-dimensional joint a posteriori probability density of I_k 's start time and duration. This density (with variables t and d) corresponds to the a posteriori probability that A_k starts at time t (i.e. $st_k=t$) and has duration d (i.e. $du_k=d$) and that the temporal relation constraints C_1, C_2, \dots, C_m are all satisfied.

In order to avoid the accumulation of parentheses in iterated integrals, we adopt the usual convention that:

$$\int_A^B f(\xi) d\xi \int_{L(\xi)}^{U(\xi)} h(\eta) d\eta \text{ denotes } \int_A^B [f(\xi) \int_{L(\xi)}^{U(\xi)} h(\eta) d\eta] d\xi$$

We will also be using the following functions:

- $\alpha(\text{predicate})$ is a function that returns 1 when *predicate* evaluates to true and 0 otherwise¹².
- $\beta^1[EQ_1(\xi)]$, where $EQ_1(\xi)$ is a linear equation in ξ , is a distribution¹³ such that:

$$\begin{aligned} \int_L^U \beta^1[EQ_1(\xi)] g(\xi) d\xi \\ = \alpha[L < x < U] g(x) \text{ if } x \text{ is the unique solution to } EQ_1(\xi), \\ = \int_L^U g(\xi) d\xi \text{ if } EQ_1(\xi) \text{ holds for } \forall \xi, \text{ and} \\ = 0 \text{ if } EQ_1(\xi) \text{ is inconsistent.} \end{aligned}$$

This simply expresses that the integration variable ξ is not only restricted to values between L and U but that the values it can take should also satisfy the linear equation $EQ_1(\xi)$.

- More generally $\beta^l[EQ_1(\xi_1, \dots, \xi_n), \dots, EQ_l(\xi_1, \dots, \xi_n)]$, where $EQ_i(\xi_1, \dots, \xi_n)$ (for $i=1$ to l) is a linear equation, is a distribution such that:

$$\begin{aligned} \int_{L_1}^{U_1} d\xi_1 \int_{L_2}^{U_2} d\xi_2 \dots \int_{L_n}^{U_n} \beta^l[EQ_1(\xi_1, \dots, \xi_n), \dots, EQ_l(\xi_1, \dots, \xi_n)] g[\xi_1, \dots, \xi_n] d\xi_n \\ = \int_{L_k}^{U_k} d\xi_k \int_{L_{k+1}}^{U_{k+1}} d\xi_{k+1} \dots \int_{L_n}^{U_n} \alpha[L_1 < F^1(\xi_k, \xi_{k+1}, \dots, \xi_n) < U_1] \dots \alpha[L_{k-1} < F^{k-1}(\xi_k, \xi_{k+1}, \dots, \xi_n) < U_{k-1}] \\ g[F^1(\xi_k, \xi_{k+1}, \dots, \xi_n), \dots, F^{k-1}(\xi_k, \xi_{k+1}, \dots, \xi_n), \xi_k, \xi_{k+1}, \dots, \xi_n] d\xi_n \end{aligned}$$

if the system of linear equations is consistent and equivalent to:

$$\begin{aligned} \xi_1 &= F^1(\xi_k, \xi_{k+1}, \dots, \xi_n) \\ \xi_2 &= F^2(\xi_k, \xi_{k+1}, \dots, \xi_n) \\ &\vdots \\ \xi_{k-1} &= F^{k-1}(\xi_k, \xi_{k+1}, \dots, \xi_n) \quad (k-1 \leq l) \end{aligned}$$

=0 if the system of equations is inconsistent.

2.4.2 Propagation in an Acyclic TCG with Fixed-duration Activities

In this subsection we express the a posteriori probability density $P(st_0 = t \& C_1 \& C_2 \& \dots \& C_m)$ for the start time of an arbitrary time interval I_0 in terms of the a priori start time distributions¹⁴. We assume fixed-duration activities arranged in an acyclic TCG.

We denote by $I_1^0, I_2^0, \dots, I_{p_0}^0$ the intervals directly adjacent to I_0 in the TCG (Figure 2-12).

¹²The reader who is not familiar with this formalism can think of it as a convenient way of expressing IF-statements in mathematical formulas.

¹³Our β distribution is a variation of the Dirac distribution. The reader who is not familiar with this formalism can simply look at it as a convenient way of expressing constraints on the values that an integration variable can take.

¹⁴ I_0 is an arbitrary element of $\{I_1, I_2, \dots, I_n\}$.

C_i^0 ($1 \leq i \leq p_0$) is the temporal constraint between I_0 and I_i^0 . Each time interval I_i^0 ($1 \leq i \leq p_0$) is itself related directly or indirectly to some other time intervals by a set of constraints S_i^0 . The sets S_i^0 are disjoint as the TCG is assumed to be acyclic (Figure 2-12).

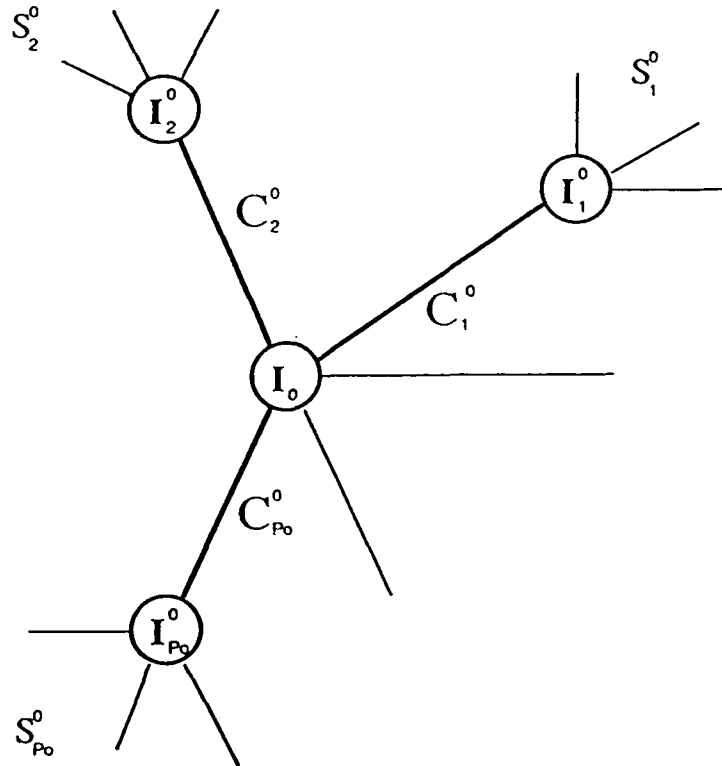


Figure 2-12: An Acyclic TCG.

The time interval I_0 is related to $I_1^0, I_2^0, \dots, I_{p_0}^0$
by respectively $C_1^0, C_2^0, \dots, C_{p_0}^0$.

Since the TCG may be disconnected, which is the case when there are several independent orders to schedule, we have:

$$\{C_1^0\} \cup S_1^0 \cup \{C_2^0\} \cup S_2^0 \dots \cup \{C_{p_0}^0\} \cup S_{p_0}^0 \subseteq \{C_1, C_2, \dots, C_m\} \quad (1)$$

and $m \leq n - 1$ (number of edges in an n -vertex tree).

We will express the (a posteriori) probability that I_0 starts at time t and that the constraints C_1, C_2, \dots, C_m are satisfied in terms of

- the a priori probability that I_0 starts at t , and
- the probabilities that each time interval I_i^0 ($1 \leq i \leq p_0$) has a start time compatible with C_i^0 given that $st_0 = t$.

These latter probabilities can be expressed in a similar fashion, thereby resulting in an inductive formulation of the a posteriori start time probabilities. The inductive formulation process stops

when all related time intervals have been accounted for (or more precisely their a priori start time distributions). At that point we have an expression of the a posteriori start time distribution that only contains a priori start time distributions, i.e. distributions that we know from the previous section.

Indeed, the a posteriori probability that $st_0=t$ and that the temporal relation constraints C_1, C_2, \dots, C_m are satisfied is given by¹⁵ the a priori probability that I_0 starts at time t , denoted $\sigma_0(st_0=t)$, multiplied by the conditional probability that C_1, C_2, \dots, C_m are satisfied, given that $st_0=t$, denoted $P(C_1 \& C_2 \& \dots \& C_m | st_0=t)$:

$$P(st_0=t \& C_1 \& C_2 \& \dots \& C_m) = \sigma_0(st_0=t) \times P(C_1 \& C_2 \& \dots \& C_m | st_0=t) \quad (2)$$

with t being the distribution variable.

Furthermore, assuming I_0 's start time fixed at time t , the satisfaction of the constraints $\{C_i^0\} \cup S_i^0$ is independent of the satisfaction of the constraints $\{C_j^0\} \cup S_j^0$ (for $i \neq j$), since we are dealing with an acyclic TCG. Hence:

$$P(C_1 \& C_2 \& \dots \& C_m | st_0=t) = \prod_{i=1}^{p_0} P(C_i^0 \& S_i^0 | st_0=t) \quad (3)$$

where $P(C_i^0 \& S_i^0 | st_0=t)$ is the conditional probability distribution that C_i^0 and the constraints in S_i^0 are satisfied given that $st_0=t$ (with t being the distribution variable).

Using (3) we can now account separately for each constraint C_i^0 . We will express each multiplicand, $P(C_i^0 \& S_i^0 | st_0=t)$, in terms of $P(st_i^0=\tau \& S_i^0)$, the probability that I_i^0 starts at some time τ (to be defined) and that the constraints in S_i^0 are satisfied. Consequently equations (2) and (3) will enable us to express $P(st_0=t \& C_1 \& C_2 \& \dots \& C_m)$ in terms of probabilities of the form $P(st_i^0=\tau \& S_i^0)$, i.e. probabilities of the same form as the original probability $P(st_0=t \& C_1 \& C_2 \& \dots \& C_m)$ except that S_i^0 is only a subset of $\{C_1, C_2, \dots, C_m\}$. By recursively repeating this process, we will be able to account for all the temporal relation constraints. The recursion process stops when S_i^0 gets empty since at that point $P(st_i^0=\tau \& S_i^0) = P(st_i^0=\tau) = \sigma_i^0(st_i^0=\tau)$, I_i^0 's a priori start time density.

Paragraphs 4.2.1 and 4.2.2 develop the computation of $P(C_i^0 \& S_i^0 | st_0=t)$ in terms of $P(st_i^0=\tau \& S_i^0) = \sigma_i^0(st_i^0=\tau)P(S_i^0 | st_i^0=\tau)$ in the case where C_i^0 is respectively of the form " I_0 MEETS I_i^0 ", and " I_0 BEFORE I_i^0 ". The treatment of the set of all thirteen of Allen's temporal relation constraints can be found in appendix 1.

2.4.2.1 C_i^0 : I_0 MEETS I_i^0

The constraint " I_0 MEETS I_i^0 " requires (Figure 2-13) that I_0 's end time be equal to I_i^0 's start time, i.e. it requires that $et_0=st_0+du_0=st_i^0$. In other words, assuming that $st_0=t$, the probability that C_i^0 and the constraints in S_i^0 are satisfied is equal to the probability that $st_i^0=t+du_0$ and that the

¹⁵ $P(A \& B) = P(A) \times P(B|A)$: the joint probability of two events A and B can be expressed as the product of the probability of event A with the conditional probability that B occurs given that A is assumed to occur.

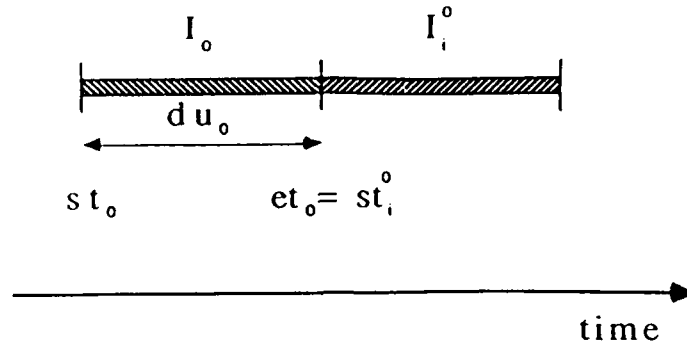


Figure 2-13: I_0 MEETS I_i^0

constraints in S_i^0 are satisfied:

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= P(st_i^0 = t + du_0 \& S_i^0) \\ &= \sigma_i^0(st_i^0 = t + du_0) P(S_i^0 | st_i^0 = t + du_0) \end{aligned} \quad (4)$$

2.4.2.2 C_i^0 : I_0 is BEFORE I_i^0

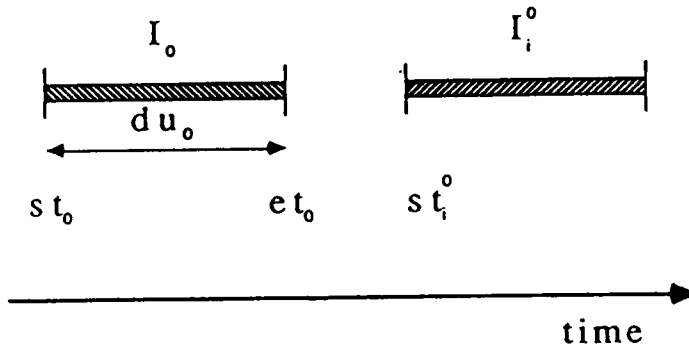


Figure 2-14: I_0 BEFORE I_i^0

The constraint " I_0 BEFORE I_i^0 " requires (Figure 2-14) that I_0 's end time be smaller than I_i^0 's start time, i.e. it requires that $et_0 = st_0 + du_0 < st_i^0$. In other words, assuming that $st_0 = t$, the probability that C_i^0 and the constraints in S_i^0 are satisfied is equal to the probability that $st_i^0 > t + du_0$ and that the constraints in S_i^0 are satisfied:

$$P(C_i^0 \& S_i^0 | st_0 = t) = \int_{t + du_0}^{\infty} P(st_i^0 = \tau \& S_i^0) d\tau$$

$$= \int_{t+du_0}^{\infty} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau) d\tau \quad (5)$$

2.4.2.3 Example

We use the three activities of order₂ (Figure 2-2) to illustrate the computations that we have just developed. We have:

$$\begin{aligned} P(st_6=t &\& C_1 \& C_2 \& \dots \& C_7) \\ &= P(st_6=t \& C_6 \& C_7) \\ &= \sigma_6(st_6=t) P(C_6 \& C_7 | st_6=t) \\ &= \sigma_6(st_6=t) \int_{t+du_6}^{\infty} \sigma_7(st_7=\tau_7) P(C_7 | st_7=\tau_7) d\tau_7 \\ &= \sigma_6(st_6=t) \int_{t+du_6}^{\infty} \sigma_7(st_7=\tau_7) d\tau_7 \int_{\tau_7+du_7}^{\infty} \sigma_8(st_8=\tau_8) d\tau_8 \end{aligned} \quad (6)$$

Also, using the formula given in appendix 1 to account for "I₇ AFTER I₆":

$$\begin{aligned} P(st_7=t &\& C_1 \& C_2 \& \dots \& C_7) \\ &= P(st_7=t \& C_6 \& C_7) \\ &= \sigma_7(st_7=t) P(C_6 \& C_7 | st_7=t) \\ &= \sigma_7(st_7=t) P(C_6 | st_7=t) P(C_7 | st_7=t) \\ &= \sigma_7(st_7=t) \int_{-\infty}^{t-du_6} \sigma_6(st_6=\tau_6) d\tau_6 \int_{t+du_7}^{\infty} \sigma_8(st_8=\tau_8) d\tau_8 \end{aligned} \quad (7)$$

Finally, in the same fashion:

$$\begin{aligned} P(st_8=t &\& C_1 \& C_2 \& \dots \& C_7) \\ &= P(st_8=t \& C_6 \& C_7) \\ &= \sigma_8(st_8=t) P(C_6 \& C_7 | st_8=t) \\ &= \sigma_8(st_8=t) \int_{-\infty}^{t-du_7} \sigma_7(st_7=\tau_7) P(C_6 | st_7=\tau_7) d\tau_7 \\ &= \sigma_8(st_8=t) \int_{-\infty}^{t-du_7} \sigma_7(st_7=\tau_7) d\tau_7 \int_{\tau_7+du_6}^{\infty} \sigma_6(st_6=\tau_6) d\tau_6 \end{aligned} \quad (8)$$

We assume that the resources are initially free (i.e. no priori resource reservations). Therefore, since activities A₆, A₇, and A₈ have uniform start time utility functions, their a priori start time densities are uniform as well, and span between times 0 and 120. Figure 2-15 displays the a posteriori start time densities computed using these a priori densities. Since in this case the start time utilities are uniform, the most preferable start times for each activity are the ones that leave the most freedom to the other activities for satisfying the temporal constraints C₆ and C₇. For instance, A₆'s a posteriori start time density indicates that A₆ should start as early as possible in order to leave as much room as possible to A₇ and A₈¹⁶. As we will see in the example of

¹⁶It is important at this point to bear in mind that we have not yet accounted for resource capacity constraints. In particular we do not know the effects that these constraints will have on the domain of possible start times for A₇ and A₈. Hence, at this point, the best start times are the least committing ones with respect to the temporal constraints.

subsection 4.4, the propagation of nonuniform start time utility functions such as the ones of the activities in order₁ are influenced by a second factor. In addition to looking for start times that leave a lot of slack to the other activities that have not been scheduled yet, the propagation of nonuniform utilities gives a higher preference to higher utilities. The a posteriori distributions thereby reflect a compromise between the utilities to optimize and the need to leave enough room for selecting good start times for the other activities that have not yet been scheduled.

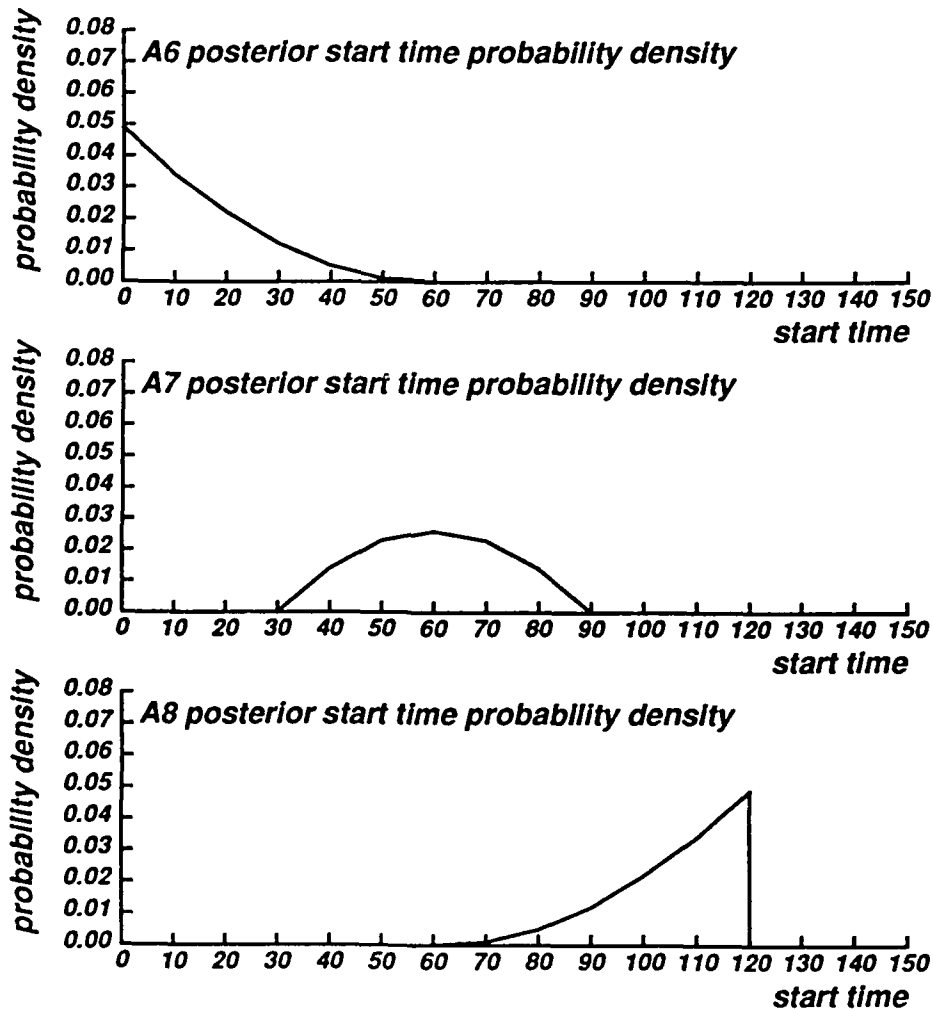


Figure 2-15: A posteriori start time densities for order₂

2.4.3 Propagation in an Acyclic TCG with Variable-duration Activities

In the case of variable-duration activities, one can compute for each activity a two-dimensional joint a posteriori probability density of the activity's start time and duration. This density, of the form $P(st_0=t \& du_0=d \& C_1 \& C_2 \& \dots \& C_m)$, represents the probability that $st_0=t$ and $du_0=d$ and that all the temporal relation constraints are satisfied given the activities' a priori start time and

duration distributions. The integrals involved in the computation of these distributions are very similar to those of the previous subsection, except that we now have to account for the a priori duration distributions.

The equivalent to equations (2) and (3) are:

$$P(st_0 = t \& du_0 = d \& C_1 \& C_2 \& \dots \& C_m) = \sigma(st_0 = t) \times \delta_0(du_0 = d) \times P(C_1 \& C_2 \& \dots \& C_m | st_0 = t \& du_0 = d) \quad (9)$$

with:

$$P(C_1 \& C_2 \& \dots \& C_m | st_0 = t \& du_0 = d) = \prod_{i=1}^{p_0} P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \quad (10)$$

Paragraphs 4.3.1 and 4.3.2 develop the computation of $P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d)$ in terms of $P(st_i^0 = \tau \& du_i^0 = \delta \& S_i^0) = \sigma_i^0(st_i^0 = \tau) \delta_i^0(du_i^0 = \delta) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta)$ in the case where C_i^0 is respectively of the form " I_0 MEETS I_i^0 ", and " I_0 BEFORE I_i^0 ". The treatment of the set of all thirteen of Allen's temporal relation constraints can be found in appendix 1. Exactly like in the previous subsection, one can use these equations in a recursive fashion to express the a posteriori start time and duration densities in terms of the a priori ones.

2.4.3.1 C_i^0 : I_0 MEETS I_i^0

As before, the constraint " I_0 MEETS I_i^0 " requires (Figure 2-13) that I_0 's end time be equal to I_i^0 's start time, i.e. it requires that $et_0 = st_0 + du_0 = st_i^0$. Hence, assuming that $st_0 = t$ and $du_0 = d$, the probability that C_i^0 and the constraints in S_i^0 are satisfied is equal to the probability that $st_i^0 = t + d$ and that the constraints in S_i^0 are satisfied:

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) &= P(st_i^0 = t + d \& S_i^0) \\ &= \int_0^\infty \delta_i^0(du_i^0 = \delta) \sigma_i^0(st_i^0 = t + d) P(S_i^0 | st_i^0 = t + d \& du_i^0 = \delta) d\delta \\ &= \int_0^\infty \delta_i^0(du_i^0 = \delta) d\delta \int_0^\infty \beta^1(\tau = t + d) \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (11)$$

The first equality is the most natural one. We will however use equation (11) in the next subsection, when allowing for cycles in the TCG.

2.4.3.2 C_i^0 : I_0 BEFORE I_i^0

In the same fashion, if I_0 has to be BEFORE I_i^0 , one has:

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_0^\infty \delta_i^0(du_i^0 = \delta) d\delta \int_{t+d}^\infty \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (12)$$

2.4.4 Relaxing the Acyclicity Assumption

We now turn our attention to the case where there may be cycles in the TCG. Equation (9) still holds but the computation of $P(C_1 \& C_2 \& \dots \& C_m | st_0 = t \& du_0 = d)$ becomes more complex.

$P(C_1 \& C_2 \& \dots \& C_m | st_0 = t \& du_0 = d)$ is the probability that C_1, C_2, \dots, C_m are satisfied when $st_0 = t$ and $du_0 = d$ given the activities' a priori start time and duration distributions. The set of temporal relation constraints $\{C_1, C_2, \dots, C_m\}$ can be expressed as a set of linear equalities and inequalities (e.g. I_i BEFORE I_j is equivalent to $st_i + du_i < st_j$). This set of equalities and inequalities together with the conditions $st_0 = t$ and $du_0 = d$ defines a polyhedron in the $2(n-1)$ dimensional space generated by $st_1^*, st_2^*, \dots, st_{n-1}^*, du_1^*, du_2^*, \dots, du_{n-1}^*$ ¹⁷. The volume contained in this polyhedron is the domain of admissible values for $st_1^*, st_2^*, \dots, st_{n-1}^*, du_1^*, du_2^*, \dots, du_{n-1}^*$ given the TCG and the conditions $st_0 = t$ and $du_0 = d$ (independently of the a priori start time and duration distributions). Therefore $P(C_1 \& C_2 \& \dots \& C_m | st_0 = t \& du_0 = d)$ can be obtained by integrating the multivariable probability density $\sigma_1^*(st_1^* = \tau_1) \sigma_2^*(st_2^* = \tau_2) \dots \sigma_{n-1}^*(st_{n-1}^* = \tau_{n-1}) \delta_1^*(du_1^* = \delta_1) \delta_2^*(du_2^* = \delta_2) \dots \delta_{n-1}^*(du_{n-1}^* = \delta_{n-1})$ over this volume. In this subsection we explain how to effectively build this multiple integral as an *iterated integral*.

Notice that, according to Fubini's theorem (see [Thomas 83] for instance), there are $[2(n-1)]!$ correct ways to express a $2(n-1)$ -tuple integral as an iterated integral (each corresponding to a permutation of the $2(n-1)$ integration variables). The algorithm that we present builds one of these $[2(n-1)]!$ iterated integrals. Although all the iterated forms are theoretically equivalent, some result in faster numerical evaluation than others¹⁸. The algorithm that we describe gives one way to build these integrals. The integrals can then be rearranged in order to speed up their evaluations. We will not be concerned here with these implementation details.

Consider again the TCG associated to order₂ in the example of subsection 1.3 (Figure 2-16)¹⁹.

As we saw in equation (6):

$$P(C_6 \& C_7 | st_6 = t) = \int_{t+du_6}^{\infty} \sigma_7(st_7 = \tau_7) d\tau_7 \int_{\tau_7+du_7}^{\infty} \sigma_8(st_8 = \tau_8) d\tau_8 \quad (13)$$

Alternatively, we can start integrating on st_8 , which produces:

$$\begin{aligned} P(C_6 \& C_7 | st_6 = t) &= \int_{-\infty}^{+\infty} \sigma_8(st_8 = \tau_8) \alpha(\tau_8 - du_7 > t + du_6) d\tau_8 \int_{t+du_6}^{\tau_8 - du_7} \sigma_7(st_7 = \tau_7) d\tau_7 \\ &= \int_{t+du_6+du_7}^{\infty} \sigma_8(st_8 = \tau_8) d\tau_8 \int_{t+du_6}^{\tau_8 - du_7} \sigma_7(st_7 = \tau_7) d\tau_7 \end{aligned} \quad (14)$$

where $\alpha(\tau_8 - du_7 > t + du_6)$ simply expresses that the second integral's upper bound has to be

¹⁷Where $\{I_0, I_1^*, I_2^*, \dots, I_{n-1}^*\} = \{I_1, I_2, \dots, I_n\}$, I_0 being an arbitrary time interval of the set, as in the previous subsections. The "*" simply indicates that the time intervals have been reordered.

¹⁸Some iterated forms are also easier to solve analytically than others.

¹⁹One of the reasons for choosing this example is that the domain of integration can be visualized in 2-D.

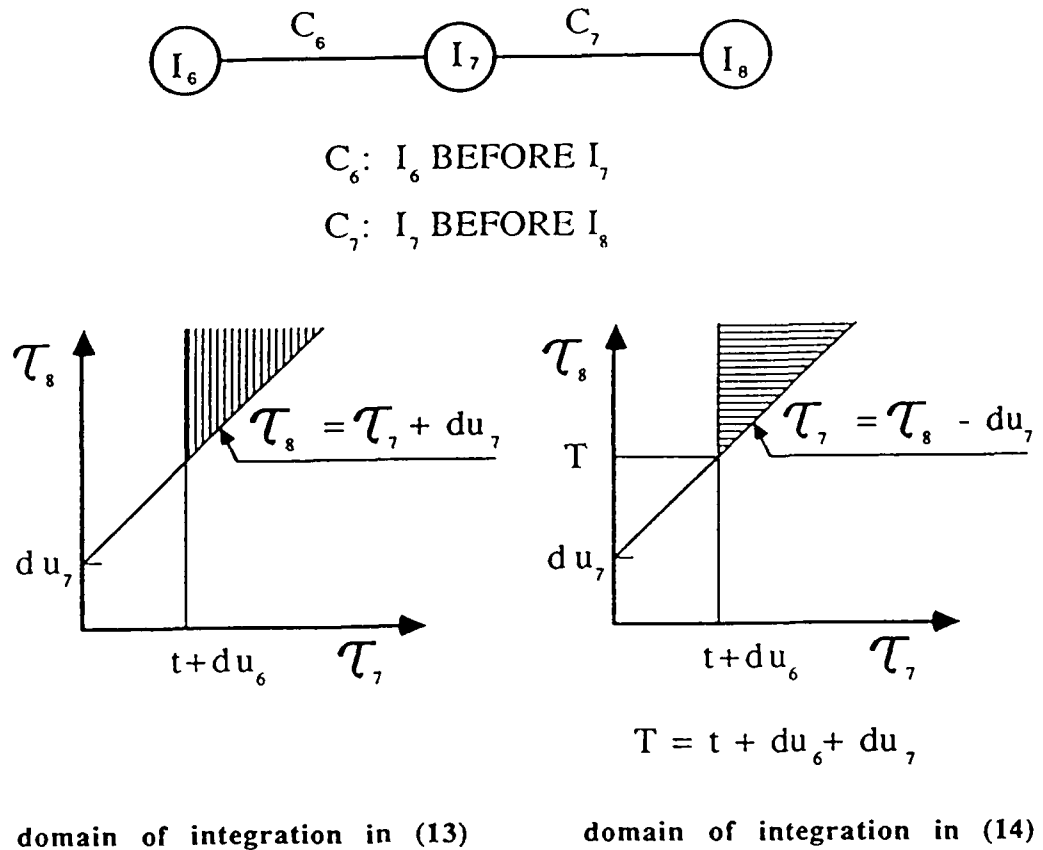


Figure 2-16: Illustration of Fubini's Theorem in a TCG with 3 Time Periods

greater than its lower bound (since we are integrating probability densities). Figure 2-16 represents the domain of integration of both form (13) and (14). They are obviously the same, which illustrates that both forms (both iterated integrals) are equivalent. Besides its illustration of Fubini's theorem, this example shows how to account for several constraints at the same time when determining a variable's domain of integration: in (14) the domain of integration of st_7 is determined by the two constraints C_6 and C_7 . In this example C_6 determines the lower bound of the integral and C_7 the upper bound. The simplicity of the formulas in the previous subsections was coming from the fact that it was possible to order the integration variables so as to account for only one constraint at a time. In TCGs with cycles it is generally not possible to find such an ordering. The domain of integration of a variable is usually determined by several constraints, some affecting the lower-bound some the upper-bound. The actual lower-bound will therefore be given by the maximum of the lower-bounds produced by each constraint (i.e. the most restrictive one) and the actual upper-bound by the minimum of the upper-bounds produced by each constraint. Additionally one has to ensure that the lower-bound is smaller than the upper-bound (see α function in (14)) since we are integrating probability densities.

Order₁ (Figure 2-2) in subsection 1.3, is an example of a TCG with cycle where the integration bounds of some start times are obtained by taking the minimum or maximum of the bounds

produced by several constraints. For instance:

$$P(C_1 \& C_2 \dots \& C_5 | st_1 = t) = \int_{t+du_1}^{\infty} \sigma_2(st_2 = \tau_2) d\tau_2 \int_{t+du_1}^{\infty} \sigma_4(st_4 = \tau_4) d\tau_4$$

$$\int_{\tau_2+du_2}^{\infty} \sigma_3(st_3 = \tau_3) d\tau_3 \int_{\text{Max}\{\tau_3+du_3, \tau_4+du_4\}}^{\infty} \sigma_5(st_5 = \tau_5) d\tau_5 \quad (15)$$

As suggested by the above example, the procedure for building iterated integrals to compute a posteriori probability distributions in TCG with cycles is just a generalization of the formulas given in the previous subsections. The main differences come from the fact that it is not possible anymore to find an ordering of the integration variables that would allow for accounting for only one temporal relation constraint at a time. We just saw how to combine the lower-bounds and upper-bounds imposed by different temporal constraints. Before describing a general procedure to effectively build a posteriori probability integrals, we still have one detail to consider. Some of the formulas given in the previous subsection include β distributions (see also appendix 1). For instance $\beta^1(\tau = t + d)$ in (11) expresses that st_i^0 should be equal to $t + d$ in order for I_0 to meet I_i^0 . β distributions provide an easy way to formally handle all temporal relation constraints in the same fashion, i.e. with integrals over the duration and start time of each time interval. β distributions allow for expressing equalities involving integration variables. When several constraints involving β distributions affect the same time interval, one has to make sure that the values for the interval's start time (or duration) that they each require are compatible. This is accomplished by using the following rule (which can easily be verified using the definition of β distributions):

$$\int_{\text{Domain}} \beta^l[EQ_1, \dots, EQ_l] \beta^m[EQ_{l+1}, \dots, EQ_{l+m}] g(\xi_1, \dots, \xi_n) d\xi_1 \dots d\xi_n$$

$$= \int_{\text{Domain}} \beta^{l+m}[EQ_1, \dots, EQ_{l+m}] g(\xi_1, \dots, \xi_n) d\xi_1 \dots d\xi_n \quad (16)$$

This is illustrated by the example below.

The TCG represented in Figure 2-17 involves 3 time intervals (with variable durations), namely I_1 , I_2 , and I_3 . The temporal relation constraints are:

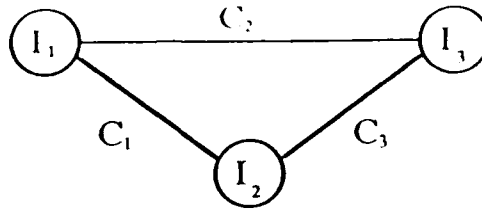
- C_1 : I_1 STARTED-BY I_2 ,
- C_2 : I_1 CONTAINS I_3 , and
- C_3 : I_2 MEETS I_3 .

Using equations (39), (41), (35) (see appendix), and (16) one can write:

$$P(C_1 \& C_2 \& C_3 | st_1 = t \& du_1 = d) = \int_0^d \delta_3(du_3 = \epsilon_3) d\epsilon_3 \int_t^{t+d-\epsilon_3} \sigma_3(st_3 = \tau_3) d\tau_3$$

$$\int_{\text{Max}\{0,0\}}^{\text{Min}\{\infty,d\}} \delta_2(du_2 = \epsilon_2) d\epsilon_2 \int_{\text{Max}\{-\infty,-\infty\}}^{\text{Min}\{+\infty,+\infty\}} \beta^2(\tau_2 = t, \tau_2 = \tau_3 - \epsilon_2) \sigma_2(st_2 = \tau_2) d\tau_2$$

This formula can be simplified using the definition of β distributions:



C_1 : I_1 STARTED-BY I_2

C_2 : I_1 CONTAINS I_3

C_3 : I_2 MEETS I_3

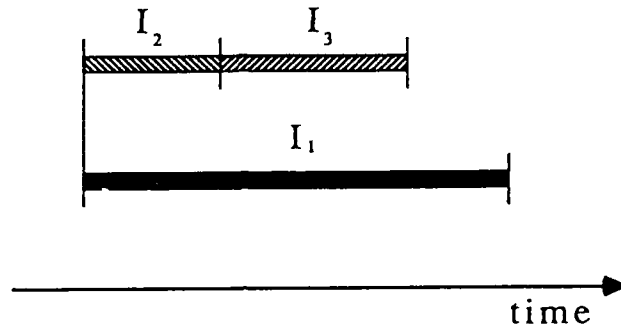


Figure 2-17: A TCG with 3 Time Periods

$$\begin{aligned}
 &P(C_1 \& C_2 \& C_3 \mid st_1=t \& du_1=d) \\
 &= \int_0^d \delta_3(du_3=\epsilon_3) d\epsilon_3 \int_t^{t+d-\epsilon_3} \sigma_3(st_3=\tau_3) \alpha(0 < \tau_3 - t < d) \delta_2(du_2=\tau_3 - t) \sigma_2(st_2=t) d\tau_3 \\
 &= \sigma_2(st_2=t) \int_0^d \delta_3(du_3=\epsilon_3) d\epsilon_3 \int_t^{t+d-\epsilon_3} \sigma_3(st_3=\tau_3) \delta_2(du_2=\tau_3 - t) d\tau_3
 \end{aligned}$$

To conclude this subsection, Figure 2-18 gives a description of BUILD-A-POSTERIORI-PROBABILITY-EXPRESSION, a general procedure to effectively express $P(C_1, \dots, C_m \mid st_0=t \& du_0=d)$ as an iterated integral. The body of the procedure makes use of a couple of simple functions, of which we only give an informal description:

- **adjacent(I , TCG)**: returns a list containing the time intervals adjacent to I in the TCG, I being itself a time interval.
- **pop(list)**: removes the first element from $list$ and returns it.
- **index(I)**: returns the index (i.e. subscript) of I , where I is a time interval (e.g. **index(I_2)** returns 2).
- **intersection(list₁, list₂)**: returns a list containing the elements of $list_1$ that are also in $list_2$ (the order of the elements in the result list is arbitrary).

- **union($list_1, list_2$):** returns a list containing any element that is either in $list_1$ or $list_2$ (or in both). An element appearing in both $list_1$ and $list_2$ is returned only once.
- **list-difference($list_1, list_2$):** returns a list with the elements of $list_1$ that are not in $list_2$.
- **start-time-upper-bound-expression($I, list$):** I is a time interval, and $list$ is a list of time intervals adjacent to I in the TCG. The function returns the start time upper-bound expression resulting from the temporal relation constraints between I and the time intervals in $list$. As explained earlier in this subsection, this is expressed as the minimum of the upper-bound produced by each constraint.
- **start-time-lower-bound-expression($I, list$):** same as above. The lower-bound is expressed as the maximum of the lower-bound produced by each constraint.
- **duration-upper-bound-expression($I, list$):** same as above for the duration (minimum).
- **duration-lower-bound-expression($I, list$):** same as above for the duration (maximum).
- **beta-expression($I, list$):** combines the eventual β distributions resulting from the temporal constraints between I and the time intervals in $list$, as explained earlier in this subsection. If there are no β distributions the function simply returns the empty expression.
- **append($expression_1, expression_2$):** appends the two expressions together.

We use ϵ and nil to respectively denote the empty expression and the empty list. The procedure builds the iterated integral from left to right by successively visiting each time interval that is directly or indirectly related to I_0 . LOCAL-EXPR contains the integrals over the start time and duration of the time interval currently visited. This local expression is appended to the right of a current partial expression of the iterated integral, thereby resulting in a new partial expression. Intervals that have been visited (i.e. whose start time and duration a priori densities have already been integrated in PARTIAL-EXPRESSION) are marked. Integration bounds for a time interval's start time and duration are determined by the temporal relation constraints between that time interval and the adjacent time intervals that have already been marked.

As illustrated in the previous examples, the expressions produced by this procedure can be simplified using the definitions of α functions and β distributions. The integration bounds can also be refined to account for the very domain over which the probability densities are strictly positive. Finally the order of integration can be rearranged to speed up evaluation. A time complexity analysis of the method and a discussion of available methods to evaluate the integrals are given in section 6.

Figure 2-19 illustrates the operation of the procedure in the construction of the iterated integral in (15). Notice that the α expressions have been omitted as they trivially evaluate to 1.

Figure 2-20 displays the a posteriori start time densities of the activities in order₁, assuming no prior resource reservations. The start time utility functions are the ones described in subsection 1.3, triangle shaped utility functions allowing for start times between 0 and 140 with a peak in 120. One should notice the difference with the propagation of the uniform start time utilities of order₂ (Figure 2-15). For instance in the case of A_5 , the a posteriori density was not totally pushed to the right. Instead the density peaks at 130, which is a compromise between the optimal

procedure BUILD-A-POSTERIORI-PROBABILITY-EXPRESSION (I_0 , TCG)

```

INTERVALS-TO-BE-PROCESSED  $\leftarrow$  adjacent( $I_0$ , TCG);
PARTIAL-EXPRESSION  $\leftarrow$   $\epsilon$ ;
MARKED-INTERVALS  $\leftarrow$   $\{I_0\}$ ;
    *a list containing  $I_0$ *
while INTERVALS-TO-BE-PROCESSED  $\neq$  nil
    I  $\leftarrow$  pop(INTERVALS-TO-BE-PROCESSED);
    i  $\leftarrow$  index(I);
    RELATED-INTERVALS  $\leftarrow$  adjacent(I, TCG);
    INTERVALS-TO-ACCOUNT-FOR  $\leftarrow$ 
        intersection(MARKED-INTERVALS, RELATED-INTERVALS);
    SUB  $\leftarrow$  start-time-upper-bound-expression(I, INTERVALS-TO-ACCOUNT-FOR);
    SLB  $\leftarrow$  start-time-lower-bound-expression(I, INTERVALS-TO-ACCOUNT-FOR);
    DUB  $\leftarrow$  duration-upper-bound-expression(I, INTERVALS-TO-ACCOUNT-FOR);
    DLB  $\leftarrow$  duration-lower-bound-expression(I, INTERVALS-TO-ACCOUNT-FOR);
    BETA  $\leftarrow$  beta-expression(I, INTERVALS-TO-ACCOUNT-FOR);
    LOCAL-EXPR  $\leftarrow$   $\alpha(DUB > DLB) \int_{DLB}^{DUB} \delta_i(du_i = \epsilon_i) \alpha(SUB > SLB) d\epsilon_i \int_{SLB}^{SUB} BETA \sigma_i(st_i = \tau_i) d\tau_i$ ;
    PARTIAL-EXPRESSION  $\leftarrow$  append(PARTIAL-EXPRESSION, LOCAL-EXPR);
    MARKED-INTERVALS  $\leftarrow$  union(MARKED-INTERVALS,  $\{I\}$ );
    INTERVALS-TO-BE-PROCESSED  $\leftarrow$  union(INTERVALS-TO-BE-PROCESSED,
        list-difference(RELATED-INTERVALS, MARKED-INTERVALS));
while-end;
return PARTIAL-EXPRESSION;

```

Figure 2-18: Procedure to express $P(C_1, \dots, C_m | st_0 = t \& du_0 = d)$ as an iterated integral

start time (120) and the tendency of the other activities to push A_5 towards its latest start time (140) in order to have more freedom. A similar remark applies to the other four activities. It should also be noted that A_1 's a posteriori start time density between 40 and 50 and A_5 's between 90 and 100 are not zero, though very small, which unfortunately does not appear very clearly on the graphs.

initialization:

PARTIAL-EXPRESSION: ϵ
MARKED-INTERVALS: $\{I_1\}$
INTERVALS-TO-BE-PROCESSED: $\{I_2, I_4\}$

step₁:

I: I_2
PARTIAL-EXPRESSION: $\int_{t+du_1}^{\infty} \sigma_2(st_2=\tau_2) d\tau_2$
MARKED-INTERVALS: $\{I_1, I_2\}$
INTERVALS-TO-BE-PROCESSED: $\{I_4, I_3\}$

step₂:

I: I_4
PARTIAL-EXPRESSION: $\int_{t+du_1}^{\infty} \sigma_2(st_2=\tau_2) d\tau_2 \int_{t+du_1}^{\infty} \sigma_4(st_4=\tau_4) d\tau_4$
MARKED-INTERVALS: $\{I_1, I_2, I_4\}$
INTERVALS-TO-BE-PROCESSED: $\{I_3, I_5\}$

step₃:

I: I_3
PARTIAL-EXPRESSION: $\int_{t+du_1}^{\infty} \sigma_2(st_2=\tau_2) d\tau_2 \int_{t+du_1}^{\infty} \sigma_4(st_4=\tau_4) d\tau_4 \int_{\tau_2+du_2}^{\infty} \sigma_3(st_3=\tau_3) d\tau_3$
MARKED-INTERVALS: $\{I_1, I_2, I_3, I_4\}$
INTERVALS-TO-BE-PROCESSED: $\{I_5\}$

step₄:

I: I_5
PARTIAL-EXPRESSION: $\int_{t+du_1}^{\infty} \sigma_2(st_2=\tau_2) d\tau_2 \int_{t+du_1}^{\infty} \sigma_4(st_4=\tau_4) d\tau_4 \int_{\tau_2+du_2}^{\infty} \sigma_3(st_3=\tau_3) d\tau_3$
 $\int_{\text{Max}\{\tau_3+du_3, \tau_4+du_4\}}^{\infty} \sigma_5(st_5=\tau_5) d\tau_5$
MARKED-INTERVALS: $\{I_1, I_2, I_3, I_4, I_5\}$
INTERVALS-TO-BE-PROCESSED: nil

Figure 2-19: Main steps involved in the construction of (15).

Notice that the α expressions have been omitted as they trivially evaluate to 1.

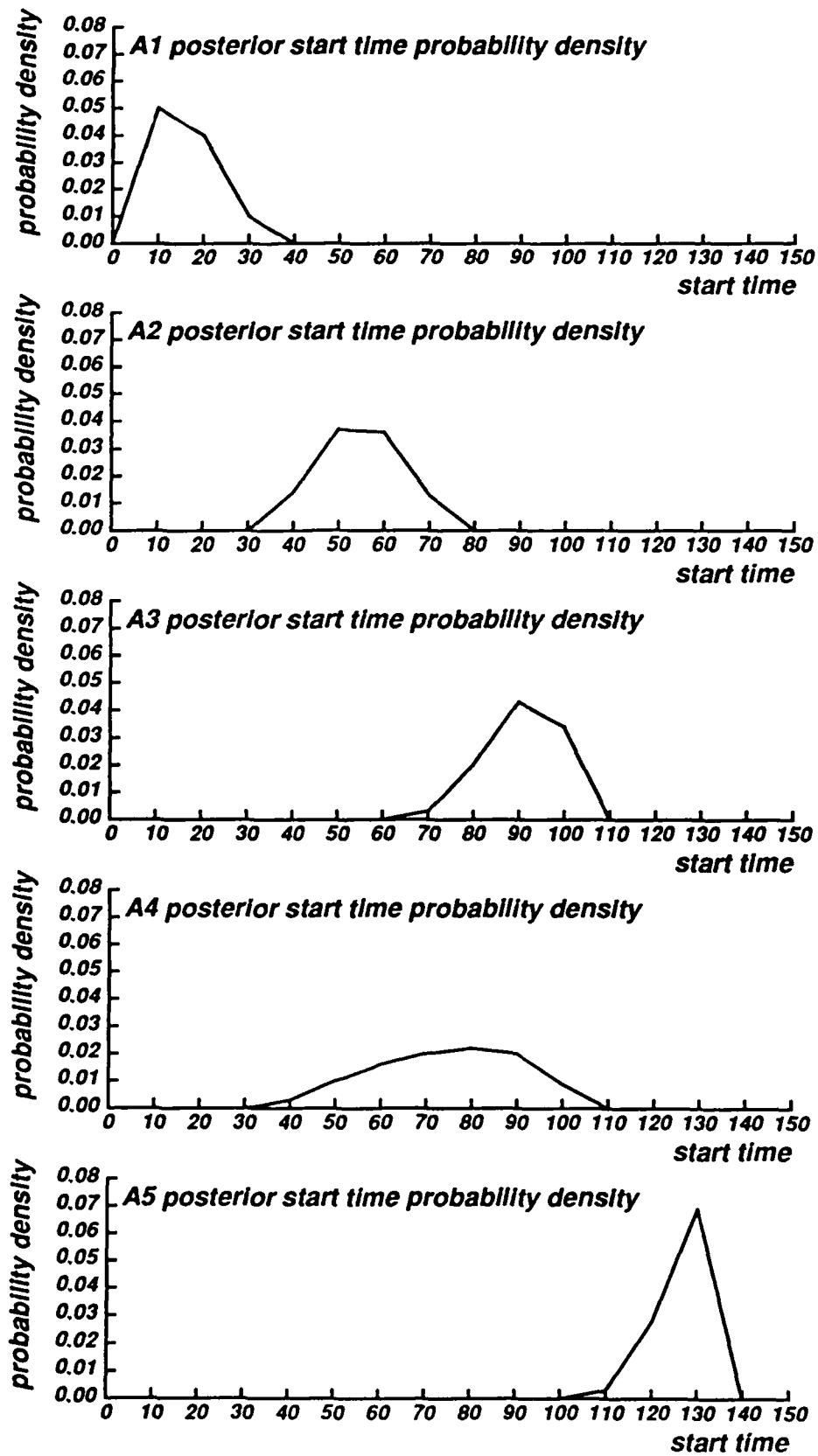


Figure 2-20: A posteriori start time densities for order₁

2.4.5 Propagation in a TCG with Explicit Disjunctions

In the case of explicit disjunctions in the TCG, one has to add the probabilities of all possible combinations of relations. Consider the example displayed in Figure 2-21. There are three time intervals I_1 , I_2 , and I_3 . The temporal relation constraints are:

- $C_1: I_1 \{ \text{BEFORE, AFTER} \} I_2$, and
- $C_2: I_2 \{ \text{BEFORE, AFTER} \} I_3$.

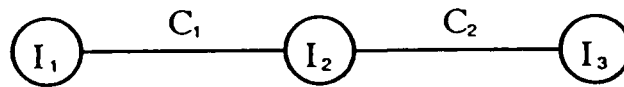
Four combinations are possible:

1. I_1 BEFORE I_2 and I_2 BEFORE I_3 ,
2. I_1 BEFORE I_2 and I_2 AFTER I_3 ,
3. I_1 AFTER I_2 and I_2 BEFORE I_3 , and
4. I_1 AFTER I_2 and I_2 AFTER I_3 .

The a posteriori probability distributions are obtained by adding the probabilities of all four possibilities. For example:

$$P(C_1 \& C_2 | st_1 = t)$$

$$\begin{aligned} &= \int_{t+du_1}^{\infty} \sigma_2(st_2=\tau_2) d\tau_2 \int_{\tau_2+du_2}^{\infty} \sigma_3(st_3=\tau_3) d\tau_3 \\ &+ \int_{t+du_1}^{\infty} \sigma_2(st_2=\tau_2) d\tau_2 \int_{-\infty}^{\tau_2-du_3} \sigma_3(st_3=\tau_3) d\tau_3 \\ &+ \int_{-\infty}^{t-du_2} \sigma_2(st_2=\tau_2) d\tau_2 \int_{\tau_2+du_2}^{\infty} \sigma_3(st_3=\tau_3) d\tau_3 \\ &+ \int_{-\infty}^{t-du_2} \sigma_2(st_2=\tau_2) d\tau_2 \int_{-\infty}^{\tau_2-du_3} \sigma_3(st_3=\tau_3) d\tau_3 \\ &= \int_{t+du_1}^{\infty} \sigma_2(st_2=\tau_2) [1 - \int_{\tau_2-du_3}^{\tau_2+du_2} \sigma_3(st_3=\tau_3) d\tau_3] d\tau_2 \\ &+ \int_{-\infty}^{t-du_2} \sigma_2(st_2=\tau_2) [1 - \int_{\tau_2-du_3}^{\tau_2+du_2} \sigma_3(st_3=\tau_3) d\tau_3] d\tau_2 \end{aligned}$$



$$C_1: I_1 \{ \text{BEFORE, AFTER} \} I_2$$

$$C_2: I_2 \{ \text{BEFORE, AFTER} \} I_3$$

Figure 2-21: A TCG with explicit disjunctions

As the number of possible combinations grows exponentially with the number of time intervals, the computations are expected to quickly become intractable. Fortunately, in the factory scheduling domain, it has been our experience that such disjunctions are extremely infrequent.

2.4.6 Result Interpretation

It is interesting at this point to look back at the desiderata and requirements identified in subsection 3.2 and check if they are satisfied. From subsection 3.2, we already know that our a priori start time distributions have been built to satisfy **requirement1**. In order to properly perform consistency checking we still need to check **requirement2**, i.e. we need to make sure that no admissible value may receive a zero a posteriori probability. This just follows from probability theory. Moreover the method will give a zero a posteriori probability to any value forbidden by the TCG, given the a priori probability distributions. Hence the computation of the a posteriori probabilities is perfect with respect to **desideratum2**, as far as the interactions defined by the TCG are concerned. The remaining inconsistencies result from the difficulty to account for inter-order interactions between unscheduled activities.

Once the a posteriori start time and duration distributions have been computed, one has to distinguish between two possible situations:

1. If at least one of the a posteriori probability density is uniformly zero then the current CSP is unsatisfiable (inconsistent). The incremental scheduler should backtrack, if still possible.
2. Otherwise, after having been normalized, the a posteriori distributions can be combined to obtain the resource demand densities induced by the CSP, as we describe in the next section²⁰. The normalization simply expresses that the total probability that each activity occurs is equal to one.

Because they account for the interactions defined by the TCG, a posteriori start time (and duration) distributions reflect intra-order interactions. They generalize the Operations Research notion of activity slack [Johnson 74]. Indeed a value with a high a posteriori probability will usually correspond to a high utility and will be likely to leave a lot of freedom for selecting high utility values for the other variables that have not been assigned a value yet. Therefore selection of start times (and durations) with high a posteriori probabilities is expected to result in good solutions to the CSP (**desideratum1**)²¹. An activity whose range of admissible start times (and durations) with high a posteriori probabilities is very wide is an activity with a lot of slack (with respect to the TCG). On the other hand, if the range of admissible values with high a posteriori probabilities is small, the activity has little slack. Equivalently, from a constraint satisfaction point of view, these a posteriori distributions can be seen locally as measures of value goodness and globally as measures of variable looseness.

However it is important to understand that, in general, the peaks of the a posteriori start time and duration distributions will not exactly coincide with the optimal activity start times and durations of the problem, nor will they even coincide with those of the problem obtained by omitting the resource capacity constraints. For instance, in the case of order₁, the optimal start times of A₁, A₂, A₃, A₄, and A₅ are respectively 30, 60, 90, 60, and 120 (and 30, 60, 90, 90, and 120 if one omits R₂'s capacity constraint). Obviously these optimal start times do not exactly coincide with the peaks of the distributions displayed in Figure 2-20. This is because at this stage

²⁰This situation is not a guarantee that the current CSP is satisfiable since we have only performed partial consistency checking.

²¹Although one should still account for inter-order interactions, which is the topic of the next section.

we have not accounted precisely for the interactions induced by the capacity constraints. Our a priori distributions accounted only implicitly for the existence of these interactions by assuming non zero probabilities for values that were not locally optimal (see Figure 2-9). Iterating the propagation process as suggested in subsection 3.2, i.e. using resource demand densities to guess new a priori probabilities, should improve the quality of the a posteriori start time and duration distributions as measures of start time and duration goodness (*desideratum1*).

2.5 Resource Demand Densities

We complete the propagation process by combining the a posteriori start time (and duration) probabilities to estimate the amount of contention for each resource. This is performed in two steps:

1. For each activity A_k , we compute a set of *individual demand densities* D_{kij} . For each resource R_{kij} that an activity A_k can use, the demand density $D_{kij}(t)$ reflects the probability that A_k uses R_{kij} at time t to fulfill its resource requirement R_{ki} . This probability depends both on the probability that A_k is active at time t and the probability that A_k uses R_{kij} to fulfill its requirement R_{ki} . The probability that an activity is active at some time t is given by the probability that the activity's start time and duration are such that the activity does not start after t and does not start so early that it is already finished by t . In the case of a fixed-duration activity A_k , this is the probability that the activity starts some time between $t-du_k$ and t . A detailed treatment of the computation of individual demand densities is given in appendix 2. We will also interpret $D_{kij}(t)$ as the reliance of A_k on the possession of R_{kij} at time t . Indeed activities with little slack and few good possible resources will have high individual demand densities concentrated over short time periods and a few resources, whereas activities with a lot of slack and several good resource alternatives will have smoother individual demand densities spread over long periods of time and several resources (see appendix 2 for details).
2. For each resource, activities' individual demand densities are combined to obtain the resource's *aggregate demand density*. This density gives the expected demand for the resource as a function of time. In the example described in subsection 1.3, the aggregate demand densities are given by:

- R_1 's aggregate demand density = $D_1(t) = D_{111}(t) + D_{211}(t) + D_{511}(t)$

- R_2 's aggregate demand density = $D_2(t) = D_{112}(t) + D_{312}(t) + D_{412}(t) + D_{712}(t)$

- R_3 's aggregate demand density = $D_3(t) = D_{613}(t) + D_{713}(t) + D_{813}(t)$

Notice that the aggregation process is performed regardless of the resources' capacities. As a matter of fact, a resource's aggregate demand density at some time t may get larger than its capacity. In general high contention for a resource will require prompt attention from the scheduler.

Figure 2-22 depicts the aggregate demand densities $D_1(t)$, $D_2(t)$, and $D_3(t)$ for the example in subsection 1.3. Clearly the contention between A_3 , A_4 and A_7 for R_2 , which was predicted in the introduction, has been identified by the propagation method. It corresponds to the peak of $D_2(t)$ centered around $t=100$. This peak reaches a density of 1.5, which is much larger than any of the other peaks. Figure 2-23 shows the individual contributions of A_3 , A_4 , and A_7 to the demand around the peak, namely $D_{312}(t)$, $D_{412}(t)$, and $D_{712}(t)$. An area of width 30 has been

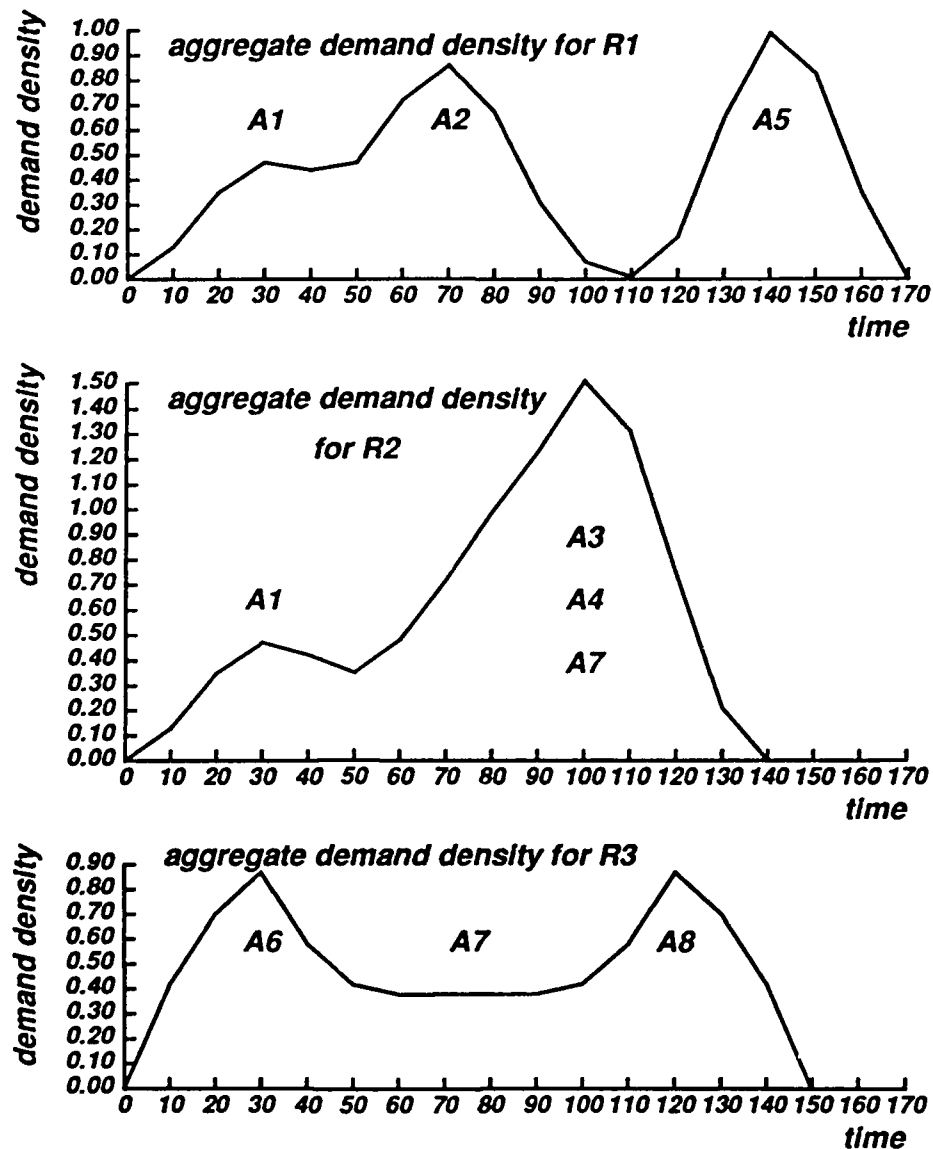


Figure 2-22: R_1 , R_2 , and R_3 's aggregate demand densities

delimited around the peak²². This is the area of high contention for R_2 . It clearly appears that within that zone, A_3 is the activity whose individual demand density contributes most to the demand for R_2 . Consequently A_3 is the activity that *relies* the most on the possession of R_2 within the area of high contention. An incremental scheduler can accordingly decide to first

²²There is no particular reason for choosing 30 except that it seems to be a characteristic duration for this problem, since all the activities have a duration of 30. The same results would hold if we were considering slightly smaller or larger intervals of contention around the peak.

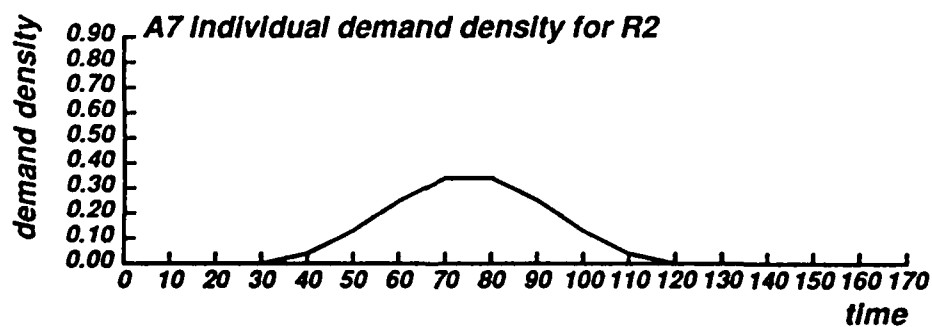
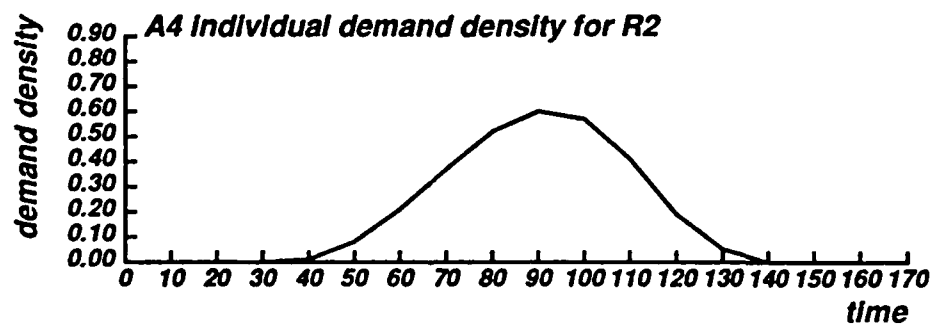
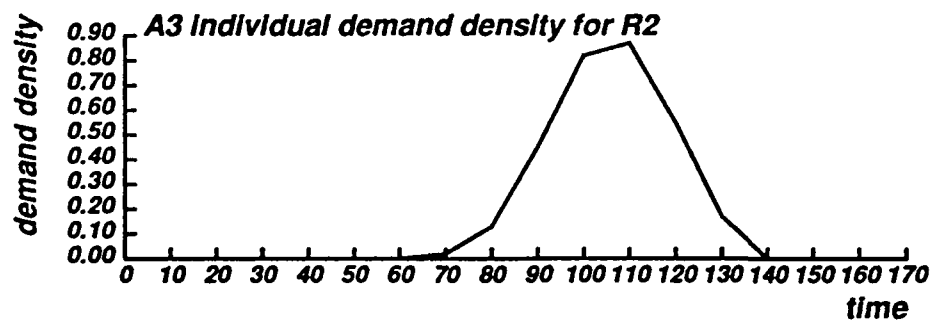
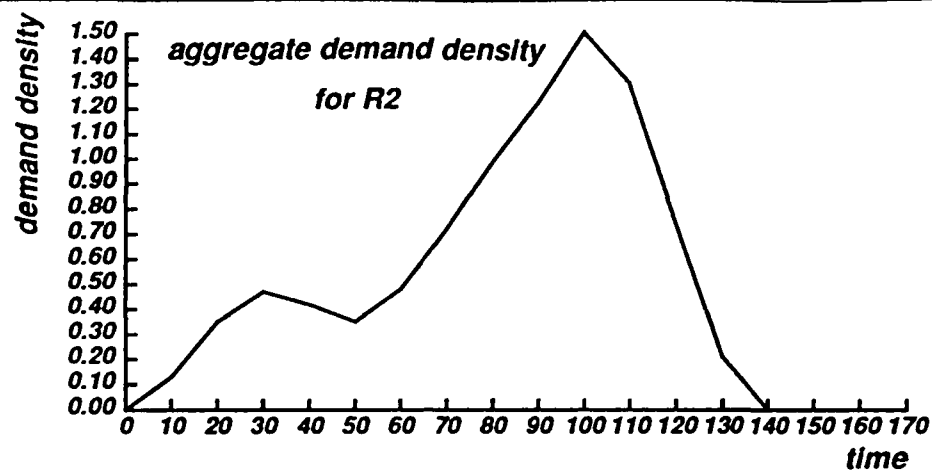


Figure 2-23: Contributions of A_3 , A_4 , and A_7 to R_2 's aggregate demand density

focus its attention on the scheduling of A_3 .

2.6 Discussion

2.6.1 Time Complexity

As demonstrated in section 4, in a TCG with no explicit disjunctive constraints, $P(st_k = t \& du_k = d \& C_1 \& C_2 \& \dots \& C_m)$ can be expressed at worst with a $2(n-1)$ -tuple integral of a priori start-time and duration probability densities, where n is the number of activities to schedule. The construction of these integrals can be performed in polynomial time (see the procedure in Figure 2-18). On the other hand, evaluation of multiple integrals using classical integration techniques requires exponential time. In the worst case computation of the n a posteriori distributions requires $O(nK^{2n})$ integrand evaluations, where K is a constant that depends on the integration method. In the case of fixed-duration activities this complexity is still $O(nK^n)$. This exponential worst-case time complexity is actually a very pessimistic one. In manufacturing environments activities are grouped in orders. Only activities within the same order have temporal relation constraints between them. Therefore the largest multiple integrals that one has to evaluate correspond to the largest number of interconnected activities within an order (say $\text{Max } m_{\text{order}}$). This results in a worst case time complexity of $nK^{2\text{Max } m_{\text{order}}}$ in the case of n variable-duration activities, and a time complexity of $nK^{\text{Max } m_{\text{order}}}$ in the case of n fixed-duration activities. This also means that, for a given set of order types (i.e. $\text{Max } m_{\text{order}}$ is fixed), the worst-case time complexity to compute the a posteriori probability distributions is *linear in the number of orders to schedule*. The computation of the resource demand densities requires at most $O(n \times r)$ steps, for n activities and r resources. Hence for a fixed set of order types and a fixed set of resources, the asymptotic time complexity of the approach is linear in the number of orders to schedule.

In manufacturing environments, one may have to schedule up to several thousands of activities grouped in orders of up to 20 or 30 activities. Assuming that all these activities are modeled as variable-duration activities, one may have multiple integrals of dimension up to 60. Numeric evaluation of such integrals is usually performed using *Monte Carlo* techniques [Stroud 71]. [Lepage 78] describes an adaptive Monte Carlo method for evaluating multidimensional integrals whose *asymptotic time and space complexities are linear in the integral's dimension*.

Alternatively one may try to reduce the size of the integrals via the use of a *hierarchical scheduler*.

2.6.2 Expressiveness of the Model

The preference propagation techniques that we have presented allow for all thirteen of Allen's temporal relation constraints as well as for disjunctions of such constraints. Additionally quantitative temporal relation constraints such as "*Activity_B should start at least 5 minutes after activity_A*" can be represented using dummy activities. For instance, one can introduce a dummy activity_C with duration of 5 minutes and the two constraints "*activity_A MEETS activity_C*" and "*activity_C BEFORE activity_B*". Using duration preferences one can express even more complex quantitative temporal relation constraints such as "*Activity_B should start as soon as possible within 5 minutes after activity_A*".

Our model accounts for three types of local preferential constraints: start time, duration, and resource preferential constraints. End time preferential constraints can be expressed using dummy activities. For instance an end time preferential constraint on an *activity_A* can be expressed as a start time preferential constraint on a dummy *activity_B* MET-BY *activity_A*. Our framework also seems to allow for the representation of the most common global organizational constraints [Baker 74]. For instance, minimizing mean (weighted) order tardiness can be expressed with the help of end time (hence start time) constraints on the last activities of each order. Minimization of mean (weighted) order flowtime can be represented with aggregate activities, each containing all the activities in an order and a preferential constraint on the duration of each aggregate activity.

2.6.3 Possible Improvements

The preference propagation technique presented in this paper has been implemented on a Sun 3/60 running Knowledge Craft on top of Lucid Common Lisp for TCGs with fixed-duration activities interconnected by BEFORE/AFTER relations and that may contain cycles. An incremental scheduler has also been built that uses the preference propagation module to focus its attention. Preliminary experimentation with the system suggests several possible improvements.

2.6.3.1 Iterative and Hierarchical Preference Propagation

As already mentioned in subsection 3.2, we are currently investigating alternative ways to compute a priori probability distributions. In particular we are considering both iterative and hierarchical variations of the preference propagation scheme presented in this paper. In an iterative approach one can use the resource demand densities obtained by the previous iteration to estimate the probability that a given resource will be available for an activity at some time t . These probabilities can then be combined to obtain more accurate start time, duration, and resource a priori probability distributions for a new propagation. For instance good start times for which good resources are likely to be unavailable would see their a priori probability being reduced. A hierarchical approach is similar except that the additional information is obtained from the results of the propagation at the upper level rather than from the previous iteration. Such techniques are expected to account more accurately for the resource requirement interactions of unscheduled activities.

2.6.3.2 Activity Criticality

In the introduction we have defined a critical activity as one whose good (overall) start times and resources are likely to become unavailable if one started scheduling other activities first. In this paper we have assumed that the most critical activity is the one that relied the most on the possession of the most contended resource (over the area of high contention for that resource). This measure of activity criticality is only concerned with the availability of good resources at good start times. Good start times may however become unavailable just because of operation precedence interactions (i.e. intra-order interactions), as reflected in the a posteriori start time/duration distributions.²³ We are looking for ways to integrate the notion of start

²³In the approach that we have presented, intra-order interactions are accounted for indirectly via the individual and aggregate demand densities, since these densities are computed from the a posteriori start time/duration distributions.

time/duration looseness identified in subsection 4.6 directly into our measure of activity criticality rather than only indirectly through measures of resource contention and activity resource reliance. Additionally, rather than simply accounting for activity reliance with respect to the most contended resource, we would like to develop a measure that accounts for the reliance of an activity on each of its possible resources and the contention on each of these resources (over the appropriate time intervals).

2.6.3.3 Value Goodness

All along we have assumed that value goodness was solely determined by the problem constraints, i.e. both the required and preferential constraints of the problem. A more sophisticated approach would consist in also accounting for the time available to come up with a schedule. If there is very little time available, one will be mainly concerned with finding an admissible schedule as soon as possible. Good values are therefore the ones that are the least likely to result in backtracking, i.e. the least constraining values identified in earlier work in constraint satisfaction with uniformly preferred values [Haralick 80]. Instead if more time is available, it may be worthwhile considering riskier values because they are likely to result in a better schedule. For instance, if one machine is more accurate than all the other ones, one could try to schedule more activities on the most accurate machine. This may however result into some extra backtracking due to the higher contention for the accurate machine.

2.7 Summary and Concluding Remarks

Factory scheduling is subject to a wide variety of preferential constraints such as meeting due dates, reducing order flowtime, using accurate machines, etc. These local a priori preferences interact. For instance, meeting an order's due date may prevent the scheduler from selecting an accurate machine for an operation. Therefore selecting start times or resources based solely on such preferences is likely to result in poor schedules. *Preference propagation* strives for the construction of measures that reflect preference interactions. Such measures can then serve to guide the construction of good *overall* schedules rather than schedules that locally optimize a subset of a priori preferences.

Our approach to preference propagation is inspired by two CSP look-ahead techniques known as variable ordering and value ordering [Dechter 88]. Both theoretical and empirical studies [Haralick 80, Freuder 82b, Nudel 83, Purdom 83, Stone 86] indicate that these techniques can significantly reduce the amount of search for a solution. Earlier work had only focused on applying these techniques to CSPs where variables have finite sets of equally preferred values. Our approach to preference propagation extends these techniques to CSPs where variables have infinite bounded sets of possible values with non-uniform preferences. The results of the propagation are formulated as a set of texture measures. In this paper we have identified the following texture measures: start time/duration goodness and looseness, resource contention, and activity resource reliance.

From an Operations Research point of view, our preference propagation technique combines advantages of both order-based and resource-based scheduling by accounting for both intra-order and inter-order interactions [Smith 85].

We perform preference propagation within a probabilistic framework. A probability is associated to each variable's possible value that dynamically reflects the likelihood that the value

results in a good schedule overall. We have identified requirements and desiderata to guide the construction of such probabilities. These requirements and desiderata have been motivated by a double objective:

1. We want to be able to detect unsatisfiable CSPs as soon as possible (quick pruning), and
2. we want to use the propagation results to help focus the scheduler's attention on the most critical decision points and the most promising decisions at these points (opportunistic scheduling).

We have argued that the approach presented in this paper fulfills these requirements and desiderata.

We have described an algorithm to perform preference propagation in T/CCGs. The algorithm deals with all thirteen of Allen's temporal relation constraints and allows for cycles in the corresponding TCG. The algorithm also allows for activity start time, duration, and resource preferences and accounts for earlier resource reservations if any. We have shown that the results of the propagation across the temporal constraints can be combined to estimate resource contention and activity resource reliance. We have also analyzed the computational requirements of our approach.

The importance of this research lies in its attempt to give a more formal characterization of the problem space, in which we carry the search for a schedule. Given the underlying uncertainty of any search problem, a probabilistic characterization is a very attractive one. In this paper, we have presented a model that uses Bayesian probabilities to account for preference interactions in T/CCGs. The problem space is finally characterized by a set of textures that are used to guide the search process.

Appendix1: A Posteriori Start Time and Duration Distributions

In this appendix we summarize the essential formulas developed in subsections 4.2 and 4.3 and complete them to allow for all 13 of Allen's temporal relation constraints. The notations are the ones defined in subsection 4.1.

0.1 Acyclic TCG with fixed-duration activities

We found in subsection 4.2 that:

$$P(st_0 = t \& C_1 \& C_2 \& \dots \& C_m) = \sigma_0(st_0 = t) \times P(C_1 \& C_2 \& \dots \& C_m | st_0 = t) \quad (17)$$

with:

$$P(C_1 \& C_2 \& \dots \& C_m | st_0 = t) = \prod_{i=1}^{P_0} P(C_i^0 \& S_i^0 | st_0 = t) \quad (18)$$

C_i^0 may be any of Allen's thirteen temporal relation constraints:

0.1.1 C_i^0 : I_0 MEETS I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= P(st_i^0 = t + du_0 \& S_i^0) \\ &= \sigma_i^0(st_i^0 = t + du_0) P(S_i^0 | st_i^0 = t + du_0) \end{aligned} \quad (19)$$

0.1.2 C_i^0 : I_0 MET-BY I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= P(st_i^0 = t - du_i^0 \& S_i^0) \\ &= \sigma_i^0(st_i^0 = t - du_i^0) P(S_i^0 | st_i^0 = t - du_i^0) \end{aligned} \quad (20)$$

0.1.3 C_i^0 : I_0 BEFORE I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= \int_{t + du_0}^{\infty} P(st_i^0 = \tau \& S_i^0) d\tau \\ &= \int_{t + du_0}^{\infty} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau) d\tau \end{aligned} \quad (21)$$

0.1.4 C_i^0 : I_0 AFTER I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= \int_{-\infty}^{t - du_i^0} P(st_i^0 = \tau \& S_i^0) d\tau \\ &= \int_{-\infty}^{t - du_i^0} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau) d\tau \end{aligned} \quad (22)$$

0.1.5 C_i^0 : I_0 DURING I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= \alpha(du_0 < du_i^0) \int_{t + du_0 - du_i^0}^t P(st_i^0 = \tau \& S_i^0) d\tau \\ &= \alpha(du_0 < du_i^0) \int_{t + du_0 - du_i^0}^t \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau) d\tau \end{aligned} \quad (23)$$

0.1.6 C_i^0 : I_0 CONTAINS I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= \alpha(du_0 > du_i^0) \int_t^{t + du_0 - du_i^0} P(st_i^0 = \tau \& S_i^0) d\tau \\ &= \alpha(du_0 > du_i^0) \int_t^{t + du_0 - du_i^0} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau) d\tau \end{aligned} \quad (24)$$

0.1.7 C_i^0 : I_0 STARTS I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= \alpha(du_0 < du_i^0) P(st_i^0 = t \& S_i^0) \\ &= \alpha(du_0 < du_i^0) \sigma_i^0(st_i^0 = t) P(S_i^0 | st_i^0 = t) \end{aligned} \quad (25)$$

0.1.8 C_i^0 : I_0 STARTED-BY I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= \alpha(du_0 > du_i^0) P(st_i^0 = t \& S_i^0) \\ &= \alpha(du_0 > du_i^0) \sigma_i^0(st_i^0 = t) P(S_i^0 | st_i^0 = t) \end{aligned} \quad (26)$$

0.1.9 C_i^0 : I_0 FINISHES I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= \alpha(du_0 < du_i^0) P(st_i^0 = t + du_0 - du_i^0 \& S_i^0) \\ &= \alpha(du_0 < du_i^0) \sigma_i^0(st_i^0 = t + du_0 - du_i^0) P(S_i^0 | st_i^0 = t + du_0 - du_i^0) \end{aligned} \quad (27)$$

0.1.10 C_i^0 : I_0 FINISHED-BY I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= \alpha(du_0 > du_i^0) P(st_i^0 = t + du_0 - du_i^0 \& S_i^0) \\ &= \alpha(du_0 > du_i^0) \sigma_i^0(st_i^0 = t + du_0 - du_i^0) P(S_i^0 | st_i^0 = t + du_0 - du_i^0) \end{aligned} \quad (28)$$

0.1.11 C_i^0 : I_0 OVERLAPS I_i^0

$$\begin{aligned} P(C_i^0 \& S_i^0 | st_0 = t) &= \int_{\text{Max}\{t, t + du_0 - du_i^0\}}^{t + du_0} P(st_i^0 = \tau \& S_i^0) d\tau \\ &= \int_{\text{Max}\{t, t + du_0 - du_i^0\}}^{t + du_0} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau) d\tau \end{aligned} \quad (29)$$

0.1.12 C_i^0 : I_0 OVERLAPPED-BY I_i^0

$$\begin{aligned}
 P(C_i^0 \& S_i^0 | st_0 = t) &= \int_{t - du_i^0}^{Min\{t, t + du_0 - du_i^0\}} P(st_i^0 = \tau \& S_i^0) d\tau \\
 &= \int_{t - du_i^0}^{Min\{t, t + du_0 - du_i^0\}} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau) d\tau
 \end{aligned} \tag{30}$$

0.1.13 C_i^0 : I_0 EQUALS I_i^0

$$\begin{aligned}
 P(C_i^0 \& S_i^0 | st_0 = t) &= \alpha(du_0 = du_i^0) P(st_i^0 = t \& S_i^0) \\
 &= \alpha(du_0 = du_i^0) \sigma_i^0(st_i^0 = t) P(S_i^0 | st_i^0 = t)
 \end{aligned} \tag{31}$$

0.2 Acyclic TCG with variable-duration activities

We found in subsection 4.3 that:

$$\begin{aligned}
 P(st_0 = t \& du_0 = d \& C_1 \& C_2 \& \dots \& C_m) &= \sigma(st_0 = t) \times \delta_0(du_0 = d) \times \\
 P(C_1 \& C_2 \& \dots \& C_m | st_0 = t \& du_0 = d)
 \end{aligned} \tag{32}$$

with:

$$P(C_1 \& C_2 \& \dots \& C_m | st_0 = t \& du_0 = d) = \prod_{i=1}^{p_0} P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \tag{33}$$

C_i^0 may be any of Allen's thirteen temporal relation constraints:

0.2.1 C_i^0 : I_0 MEETS I_i^0

$$\begin{aligned}
 P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) &= P(st_i^0 = t + d \& S_i^0) \\
 &= \int_0^\infty \delta_i^0(du_i^0 = \delta) \sigma_i^0(st_i^0 = t + d) P(S_i^0 | st_i^0 = t + d \& du_i^0 = \delta) d\delta \\
 &= \int_0^\infty \delta_i^0(du_i^0 = \delta) d\delta \int_{-\infty}^{+\infty} \beta^1(\tau = t + d) \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau
 \end{aligned} \tag{34}$$

The first equality is the most useful one. However equation (34) is useful for the treatment of TCGs with cycles (see subsection 4.4). The same remark applies to the other equations involving β distributions.

0.2.2 C_i^0 : I_0 MET-BY I_i^0

$$\begin{aligned}
 P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\
 &= \int_0^\infty \delta_i^0(du_i^0 = \delta) \sigma_i^0(st_i^0 = t - \delta) P(S_i^0 | st_i^0 = t - \delta \& du_i^0 = \delta) d\delta
 \end{aligned}$$

$$= \int_0^\infty \delta_i^0(du_i^0 = \delta) d\delta \int_{-\infty}^{+\infty} \beta^1(\tau = t - \delta) \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \quad (35)$$

0.2.3 C_i^0 : I_0 BEFORE I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_0^\infty \delta_i^0(du_i^0 = \delta) d\delta \int_{t+d}^\infty \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (36)$$

0.2.4 C_i^0 : I_0 AFTER I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_0^\infty \delta_i^0(du_i^0 = \delta) d\delta \int_{-\infty}^{t-\delta} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (37)$$

0.2.5 C_i^0 : I_0 DURING I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_d^\infty \delta_i^0(du_i^0 = \delta) d\delta \int_{t+d-\delta}^t \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (38)$$

0.2.6 C_i^0 : I_0 CONTAINS I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_0^d \delta_i^0(du_i^0 = \delta) d\delta \int_t^{t+d-\delta} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (39)$$

0.2.7 C_i^0 : I_0 STARTS I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_d^\infty \delta_i^0(du_i^0 = \delta) \sigma_i^0(st_i^0 = t) P(S_i^0 | st_i^0 = t \& du_i^0 = \delta) d\delta \\ &= \int_d^\infty \delta_i^0(du_i^0 = \delta) d\delta \int_{-\infty}^{+\infty} \beta^1(\tau = t) \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (40)$$

0.2.8 C_i^0 : I_0 STARTED-BY I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_0^d \delta_i^0(du_i^0 = \delta) \sigma_i^0(st_i^0 = t) P(S_i^0 | st_i^0 = t \& du_i^0 = \delta) d\delta \end{aligned}$$

$$= \int_0^d \delta_i^0(du_i^0 = \delta) d\delta \int_{-\infty}^{+\infty} \beta^1(\tau = t) \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \quad (41)$$

0.2.9 C_i^0 : I_0 FINISHES I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_d^{+\infty} \delta_i^0(du_i^0 = \delta) \sigma_i^0(st_i^0 = t + d - \delta) P(S_i^0 | st_i^0 = t + d - \delta \& du_i^0 = \delta) d\delta \\ &= \\ & \int_d^{+\infty} \delta_i^0(du_i^0 = \delta) d\delta \int_{-\infty}^{+\infty} \beta^1(\tau = t + d - \delta) \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (42)$$

0.2.10 C_i^0 : I_0 FINISHED-BY I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_0^d \delta_i^0(du_i^0 = \delta) \sigma_i^0(st_i^0 = t + d - \delta) P(S_i^0 | st_i^0 = t + d - \delta \& du_i^0 = \delta) d\delta \\ &= \\ & \int_0^d \delta_i^0(du_i^0 = \delta) d\delta \int_{-\infty}^{+\infty} \beta^1(\tau = t + d - \delta) \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (43)$$

0.2.11 C_i^0 : I_0 OVERLAPS I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_0^{+\infty} \delta_i^0(du_i^0 = \delta) d\delta \int_{\text{Max}(t, t+d-\delta)}^{t+d} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \\ &= \int_0^d \delta_i^0(du_i^0 = \delta) d\delta \int_{t+d-\delta}^{t+d} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \\ &+ \\ & \int_d^{+\infty} \delta_i^0(du_i^0 = \delta) d\delta \int_t^{t+d} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned} \quad (44)$$

0.2.12 C_i^0 : I_0 OVERLAPPED-BY I_i^0

$$\begin{aligned} & P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\ &= \int_0^{+\infty} \delta_i^0(du_i^0 = \delta) d\delta \int_{t-\delta}^{\text{Min}(t, t+d-\delta)} \sigma_i^0(st_i^0 = \tau) P(S_i^0 | st_i^0 = \tau \& du_i^0 = \delta) d\tau \end{aligned}$$

$$\begin{aligned}
&= \int_0^d \delta_i^0(du_i^0=\delta) d\delta \int_{t-\delta}^t \sigma_i^0(st_i^0=\tau) P(S_i^0|st_i^0=\tau \& du_i^0=\delta) d\tau \\
&+ \\
&\int_d^\infty \delta_i^0(du_i^0=\delta) d\delta \int_{t-\delta}^{t+d-\delta} \sigma_i^0(st_i^0=\tau) P(S_i^0|st_i^0=\tau \& du_i^0=\delta) d\tau
\end{aligned} \tag{45}$$

0.2.13 C_i^0 : I_0 EQUALS I_i^0

$$\begin{aligned}
&P(C_i^0 \& S_i^0 | st_0 = t \& du_0 = d) \\
&= \delta_i^0(du_i^0=d) \sigma_i^0(st_i^0=t) P(S_i^0 | st_i^0=t \& du_i^0=d) \\
&= \int_0^\infty \delta_i^0(du_i^0=\delta) d\delta \int_{-\infty}^{+\infty} \beta^2(\delta=d, \tau=t) \sigma_i^0(st_i^0=\tau) P(S_i^0 | st_i^0=\tau \& du_i^0=\delta) d\tau
\end{aligned} \tag{46}$$

Appendix2: Activity Individual Demand Densities

0.1 Notations

In this appendix we assume that a posteriori start time densities have already been computed as described in section 4. We assume that none of these densities is uniformly zero, otherwise this would indicate unsatisfiability of the current CSP and the incremental scheduler would have to backtrack. As already mentioned earlier, a posteriori start time densities can then be normalized to express the fact that each activity will occur once (i.e. each activity will start exactly once). These normalized a posteriori densities will be denoted:

- fixed-duration activities: $P_N(st_i=t \& C_1 \& \dots \& C_m)$
- variable-duration activities: $P_N(st_i=t \& du_i=d \& C_1 \& \dots \& C_m)$

$\rho_{R_{ki}}(R_{kij})$ will denote the a priori probability that A_k uses R_{kij} to fulfill its resource requirement R_{ki} . $D_{kij}(t)$ will denote A_k 's individual demand for R_{kij} as a function of t . This is the probability that A_k uses R_{kij} at time t to fulfill its resource requirement R_{ki} . The computations are performed assuming an incremental scheduler whose earlier resource reservations are non-preemptible. Therefore the demand density has to be reshaped so that it does not overlap with earlier reservations. We propose a method for doing so, which involves two steps.

Finally we will be using the predicate $AVAIL(R_{kij}, t, t+du_k)$ which returns **true** if and only if resource R_{kij} is available at all time between t and $t+du_k$. This is a precondition for scheduling activity A_k to start at time t , if A_k is to use resource R_{kij} .

0.2 Resource Demand Densities Produced by Fixed-Duration Activities

The probability that activity A_k uses R_{kij} at time t to fulfill its resource requirement R_{ki} is given by the a priori probability that A_k uses R_{kij} to fulfill R_{ki} multiplied by the conditional probability that A_k is active at time t given that it uses R_{kij} to fulfill R_{ki} . It turns out that this latter conditional probability may be uniformly zero for some resources R_{kij} due to earlier reservations. This can be accounted for by refining the a priori probabilities $\rho_{R_{ki}}(R_{kij})$. $D_{kij}(t)$ is therefore computed in two steps:

1. In the first step we compute:

$$D_{kij}^{step1}(t) = \rho_{R_{ki}}(R_{kij}) \int_{t-du_k}^t P_N(st_k=\tau \& C_1 \& \dots \& C_m | R_{ki}=R_{kij}) d\tau$$

where $P_N(st_k=\tau \& C_1 \& \dots \& C_m | R_{ki}=R_{kij})$ is the probability that $st_k=\tau$ and that the temporal relation constraints C_1, \dots, C_m are satisfied given the activities' a priori start time distributions and given that R_{kij} is the resource used to fulfill requirement R_{ki} . This probability can be approximated by computing I_k 's a posteriori start time distribution starting from an a priori start time distribution that accounts for R_{kij} 's reservations. We do so by replacing $\sigma_k(st_k=t)$ with $\sigma_k(st_k=t) \times \alpha[AVAIL(R_{kij}, t, t+du_k)]$ in the computation of $P(st_k=t \& C_1 \& \dots \& C_m)$. In other words:

$$P_N(st_k=\tau \& C_1 \& \dots \& C_m | R_{ki}=R_{kij}) = \kappa P(st_k=\tau \& C_1 \& \dots \& C_m) \alpha[AVAIL(R_{kij}, \tau, \tau+du_k)]$$

where κ is a normalization factor²⁴.

2. In the second step the a priori probabilities $\rho_{R_{ki}}(R_{kij})$ are refined to account for the resource reservations. The refined probabilities are denoted $\rho_{R_{ki}}^{\text{step2}}(R_{kij})$. Indeed, due to earlier reservations, some resources R_{kij} have a posteriori probabilities $P_N(st_k=\tau \& C_1 \& \dots \& C_m | R_{ki}=R_{kij})$ that are uniformly zero. A_k 's individual demand for these resources is therefore uniformly zero as well. Hence one can use the new probabilities:

$$\rho_{R_{ki}}^{\text{step2}}(R_{kij}) = \begin{cases} 0 & \text{if } P_N(st_k=\tau \& C_1 \& \dots \& C_m | R_{ki}=R_{kij}) \text{ is uniformly } 0 \\ \kappa_{ki} \rho_{R_{ki}}(R_{kij}) & \text{otherwise} \end{cases}$$

where κ_{ki} is a normalization factor. Notice that, for each R_{ki} , because of the consistency checking performed after the computation of the a posteriori start time distributions, we are guaranteed at this point to have at least one resource R_{kij} such that $P_N(st_k=\tau \& C_1 \& \dots \& C_m | R_{ki}=R_{kij})$ is not uniformly zero. One can then compute:

$$D_{kij}^{\text{step2}}(t) = \rho_{R_{ki}}^{\text{step2}}(R_{kij}) \int_{t-du_k}^t P_N(st_k=\tau \& C_1 \& \dots \& C_m | R_{ki}=R_{kij}) d\tau$$

In practice it is not necessary to compute $D_{kij}^{\text{step2}}(t)$: one can just compute $P_N(st_k=\tau \& C_1 \& \dots \& C_m | R_{ki}=R_{kij})$ and check if it is uniformly zero or not.

Finally, notice that the total demand is given by:

$$\begin{aligned} \int_{-\infty}^{+\infty} D_{kij}^{\text{step2}}(\tau) d\tau &= \rho_{R_{ki}}^{\text{step2}}(R_{kij}) \int_{-\infty}^{+\infty} d\tau \int_{\tau-du_k}^{\tau} P_N(st_k=\xi \& C_1 \& \dots \& C_m | R_{ki}=R_{kij}) d\xi \\ &= \rho_{R_{ki}}^{\text{step2}}(R_{kij}) \int_{-\infty}^{+\infty} P_N(st_k=\xi \& C_1 \& \dots \& C_m | R_{ki}=R_{kij}) d\xi \int_{\xi}^{\xi+du_k} d\tau \quad (\text{using Fubini}) \\ &= \rho_{R_{ki}}^{\text{step2}}(R_{kij}) \times du_k \quad (\text{since } P_N \text{ is normalized}) \end{aligned}$$

and hence for each R_{ki} required by A_k :

$$\sum_j \int_{-\infty}^{+\infty} D_{kij}^{\text{step2}}(\tau) d\tau = du_k$$

which simply expresses that an activity A_k 's total demand for a resource R_{ki} is equal to its duration du_k . This duration has simply been distributed over time and over several resources (R_{kij}) to account for the different possible schedules of the activity.

²⁴ Again this normalization simply expresses that the activity will occur exactly once.

0.3 Resource Demand Densities Produced by Variable-Duration Activities

The computations in the case of variable-duration activities are very similar to the ones for fixed-duration activities:

1. In the first step one computes the distributions

$$P_N(st_k = \tau \& du_k = \epsilon \& C_1 \& \dots \& C_m | R_{ki} = R_{kij}) = \kappa P(st_k = \tau \& du_k = \epsilon \& C_1 \& \dots \& C_m) \alpha[AVAIL(R_{kij}, \tau, \tau + \epsilon)]$$

where κ is a normalization factor.

2. The probabilities $\rho_{R_{ki}}(R_{kij})$ are refined in the same way as for fixed-duration activities. One can then compute:

$$D_{kij}^{step2}(t) = \rho_{R_{ki}}^{step2}(R_{kij}) \int_0^\infty d\epsilon \int_{t-\epsilon}^t P_N(st_k = \tau \& du_k = \epsilon \& C_1 \& \dots \& C_m | R_{ki} = R_{kij}) d\tau$$

Lastly, using Fubini's theorem, one can check that:

$$\begin{aligned} \int_{-\infty}^{+\infty} D_{kij}^{step2}(\tau_1) d\tau_1 &= \rho_{R_{ki}}^{step2}(R_{kij}) \int_{-\infty}^{+\infty} d\tau_1 \int_0^\infty d\epsilon \int_{\tau_1-\epsilon}^{\tau_1} P_N(st_k = \tau_2 \& du_k = \epsilon \& C_1 \dots \& C_m | R_{ki} = R_{kij}) d\tau_2 \\ &= \rho_{R_{ki}}^{step2}(R_{kij}) \int_0^\infty d\epsilon \int_{-\infty}^{+\infty} d\tau_2 \int_{\tau_2}^{\tau_2+\epsilon} P_N(st_k = \tau_2 \& du_k = \epsilon \& C_1 \dots \& C_m | R_{ki} = R_{kij}) d\tau_1 \\ &= \rho_{R_{ki}}^{step2}(R_{kij}) \int_0^\infty d\epsilon \int_{-\infty}^{+\infty} \epsilon P_N(st_k = \tau_2 \& du_k = \epsilon \& C_1 \dots \& C_m | R_{ki} = R_{kij}) d\tau_2 \end{aligned}$$

Hence, for each R_{ki} required by A_k , A_k 's total demand is:

$$\sum_j \int_{-\infty}^{+\infty} D_{kij}^{step2}(\tau_1) d\tau_1 = \int_0^\infty d\epsilon \int_{-\infty}^{+\infty} \epsilon \left[\sum_j \rho_{R_{ki}}^{step2}(R_{kij}) P_N(st_k = \tau_2 \& du_k = \epsilon \& C_1 \dots \& C_m | R_{ki} = R_{kij}) \right] d\tau_2$$

which is A_k 's expected duration given the joint start time and duration probability density $\sum_j \rho_{R_{ki}}^{step2}(R_{kij}) P_N(st_k = \tau_2 \& du_k = \epsilon \& C_1 \dots \& C_m | R_{ki} = R_{kij})$.

CHAPTER 3: Activity-Based Scheduling

3.1 Introduction

We are concerned with the issue of how to opportunistically focus an incremental job shop scheduler's attention on the most critical decision points (*variable ordering* heuristics) and the most promising decisions at these points (*value ordering* heuristics) in order to reduce search and improve the quality of the resulting schedule.

So called order-based and resource-based scheduling techniques have been at the origin of several incremental scheduling algorithms. In an order-based approach, each order is considered a single decision point, i.e. orders are prioritized and scheduled one by one. In a resource-based approach, resources are rated according to their projected levels of demand. Resources are then scheduled one by one, starting with the ones that have the highest demand. Order-based scheduling has proven to be a viable paradigm in problems where slack (i.e. temporal precedence interactions) is the dominating factor. On the other hand resource-based scheduling is likely to perform better in situations where resource contention (i.e. resource requirement interactions) is critical. Neither approach is perfect. Indeed a lot of real life scheduling problems contain a mix of critical orders and critical resources. In the past few years it has become clear that in order to perform well in a wider class of problems, schedulers need the ability to opportunistically switch from one approach to the other. The OPIS [Smith 85, Ow86, Smith86a] scheduler was the first scheduler to combine both approaches. OPIS uses demand thresholds to identify bottleneck resources. Typically, when a bottleneck resource is detected, all activities requiring that resource and that have not yet been scheduled will be scheduled. When all bottleneck resources have been scheduled, OPIS switches to order-based scheduling. While in order scheduling mode, OPIS may detect the appearance of new bottleneck resources and switch back to its resource scheduling mode. Even such an approach has its shortcomings as the criticalities of the activities requiring a bottleneck resource or belonging to a critical order are not homogeneous, i.e. some of these activities may not be that critical. This consideration lead us to the investigation of a new scheduling framework where the decision points are no longer entire resources or entire orders but instead where each activity is a decision point in its own right. Within this framework activity criticality is no longer determined via a sole bottleneck resource or via the sole order to which it belongs. Instead measures of activity criticality account for both temporal precedence interactions (i.e., so-called *intra-order* interactions [Smith 85]) and resource requirement interactions (i.e., so-called *inter-order* interactions). By simultaneously accounting for both types of interactions the approach is expected to opportunistically combine advantages of both order-based and resource-based scheduling techniques.

In this chapter, we study one variable-ordering heuristic and three value-ordering heuristics for activity-based scheduling. A preliminary set of 38 scheduling problems was used to compare the performance of these heuristics. The experiments clearly indicate that the variable-ordering heuristic significantly reduces search. The comparison of the value-ordering heuristics when combined with the variable ordering heuristic suggests that a least constraining value ordering heuristic is not the only viable approach to maintain the amount of search at an acceptable level. Indeed some other heuristics produced much better schedules without significantly increasing search.

In the next section we describe our model of the job-shop scheduling problem. The following section gives an overview of the activity-based approach to scheduling that we are investigating, and introduces a probabilistic model to account for both intra-order and inter-order interactions. Section 4 presents a variable-ordering heuristic based on this probabilistic model. In section 5 we present three value-ordering heuristics: a least constraining heuristic, a hill-climbing heuristic, and an intermediate heuristic where values are rated according to the probability that they will remain available and that no better values will remain available. Preliminary experimental results are reported in section 6. Section 7 discusses these results. Section 8 contains some concluding remarks.

3.2 The Model

Formally, we will say that we have a set of N jobs (i.e. orders) to schedule. Each job has a predefined process plan that specifies a partial ordering among the activities (i.e. operations) to be scheduled. Each activity A_k ($1 \leq k \leq n$) may require one or several resources R_{ki} ($1 \leq i \leq p_k$), for each of which there may be several alternatives R_{kij} ($1 \leq j \leq q_{ki}$)²⁵. We will use st_k , et_k , and du_k to respectively denote A_k 's start time, end time, and duration.

We view the scheduling problem as a constraint satisfaction problem (CSP).

The variables of the problem are the activity start times, the resources allocated to each activity, and possibly the durations of some activities. An activity's end time is defined as the sum of its start time and duration. Each variable has a bounded (finite or infinite) set of admissible values. For instance, the start time of an activity is always restricted at one end by the order release date and at the other end by the order latest acceptable completion time²⁶ according to the durations of the activities that precede/follow the activity in the process plan.

We differentiate between two classes of constraints: activity precedence constraints and resource capacity constraints. The activity precedence constraints are the ones defined by the process plans. Our model [Sadeh88] accounts for all 13 of Allen's temporal constraints [Allen 84]. Capacity constraints restrict the number of reservations of a resource over any time interval to the capacity of that resource. For the sake of simplicity, we will assume in this paper that all resources are of unary capacity.

Additionally our model allows for preferences on activity start times and durations as well as on the resources that activities can use. Preferences are modeled with utility functions. These functions map each variable's possible values onto utilities ranging between 0 and 1. Preferences on activity start times and durations allow for the representation of organizational goals such as reducing order tardiness, or reducing work-in-process (WIP) [For 83b, Sadeh88]. Resource preferences are very useful to differentiate between functionally equivalent resources with different characteristics (e.g. difference in accuracy). In this paper we will assume that the sum of the utility functions defines a (separable) objective function to be maximized.

²⁵It is important to keep in mind that several activities may require the same resource. For instance if two activities A_1 and A_2 both require a unique resource which has to be R_1 , we have $R_{111} = R_{211} = R_1$.

²⁶This is not necessarily the order's due date.

3.3 The Approach

3.3.1 An Activity-based Scheduler

In an activity-based approach, each activity is treated as an aggregate variable, or decision point, that comprises the activity's start time, its resources, and possibly its duration. The schedule is built incrementally by iteratively selecting an activity to be scheduled and a reservation for that activity (i.e. start time, resources and possibly duration). Every time a new activity is scheduled, new constraints are added to the initial scheduling problem, and propagated. If an inconsistency is detected during propagation, the system backtracks. The process stops either when all activities have been successfully scheduled or when all possible alternatives have been tried without success.

The efficiency of such an incremental approach critically relies on the order in which activities are scheduled and on the order in which possible reservations are tried for each activity. Indeed, because job-shop scheduling is NP-hard, search for a schedule may require exponential time in the worst case. Both empirical and analytical studies of constraint satisfaction problems reported in [Haralick 80, Freuder 82b, Purdom 83, Nadel 86b, Nadel 86c, Nadel 86d, Stone 86] indicate however that, on the average, search can significantly be reduced if always focused on the most critical decision points and the most promising decisions at these points. Such techniques are often referred to [Dechter 88] as variable and value ordering heuristics.

In this paper we assume that critical activities are the ones whose good (overall) reservations are most likely to become unavailable if one were to start scheduling other activities first. In general reservations may become unavailable because of operation precedence constraints, because of resource capacity constraints, or because of combinations of both types of constraints. Clearly criticality measures are probabilistic in nature, as their computations require probabilistic assumptions on the values that will be assigned later on to each variable (i.e. the reservations that will later on be assigned to each unscheduled activity). In the next subsection we introduce a probabilistic framework that accounts for the interactions of start time, duration and resource preferences induced by both activity precedence and resource capacity constraints. We will use this model throughout the remainder of the paper to define several variable and value ordering heuristics for activity-based scheduling.

3.3.2 A Probabilistic Framework to Account for Constraint Interactions

In this subsection we outline²⁷ a probabilistic model that we will use throughout the remainder of the paper to define several variable and value ordering heuristics for activity-based job-shop scheduling. We justify the model by its ability to account for both intra-order and inter-order interactions and by its relatively low computational requirements. For the sake of simplicity, the formulas presented in this paper assume fixed duration activities. Similar formulas can be deduced when dealing with variable duration activities.

In our model a priori probability distributions are assumed for the possible start times and resources of each unscheduled activity. These probabilities are then refined to account for the

²⁷A more detailed description can be found in [Sadeh88].

interactions induced by the problem constraints (i.e. both intra-order and inter-order interactions). Finally the results of this propagation process are combined to identify critical activities and promising reservations for these activities. In their simplest form the a priori probability distributions are uniform. This amounts to assuming that, a priori, all possible reservations are equally probable. A slightly more sophisticated model consists in *biasing* the a priori distributions towards good values as defined by the utility functions [Sadeh88]. Such biased distributions are expected to account better for value ordering heuristics that give more attention to higher utility values.

Once the a priori distributions have been built, they can be refined to account for the interactions of the problem constraints. In our model, the propagation is performed in two steps. The probability distributions are first propagated within each order, thereby accounting for intra-order interactions, and then across orders to account for inter-order interactions. Accounting simultaneously for both types of interactions seem indeed very difficult as much from a theoretical point of view as from a purely computational point of view. As a matter of fact the number of ways in which a set of activities can interact is combinatorial in the number of these activities²⁸. Instead, by separately accounting for intra-order and inter-order interactions, one greatly reduces the amount of computation to be performed. The propagation results can always be further refined by iterating the propagation an arbitrary number of times.

Concretely, once the a priori distributions have been generated, our propagation process involves the following two steps:

1.
 - a. The a priori start time probability distributions are refined to account for activity precedence constraints. The resulting (a posteriori) probability distributions associate to the possible start times of each activity the probability that these start times will be tried by the scheduler and will not result in the violation of an activity precedence constraint. These a posteriori start time distributions can be normalized to express that each activity will occur exactly once, and hence will start exactly once.
 - b. For each resource requirement R_{ki} of each activity A_k , and for each resource alternative R_{kij} to fulfill R_{ki} , we compute the probabilistic demand D_{kij} of A_k for R_{kij} as a function of time. This probability is obtained using A_k 's normalized a posteriori start time distribution and the a priori probability that A_k uses R_{kij} to fulfill its requirement R_{ki} . Hence $D_{kij}(t)$ represents the probabilistic contribution of A_k to the demand for R_{kij} at time t , if activity reservations were only checked for consistency with respect to the activity precedence constraints. Later on we will refer to $D_{kij}(t)$ as A_k 's (probabilistic) individual demand for R_{kij} at time t .
2. Finally the individual demand densities of all activities are aggregated (i.e. summed at each point in time) to reflect the probabilistic demand for each resource

²⁸In any realistic problem, Monte Carlo simulation would indeed require tremendous amounts of computations if one were to simultaneously account for all the activities and all the constraints. This is because the probability of randomly generating a schedule for all the activities, that satisfy all activity precedence and resource capacity constraints, is in general extremely small.

in function of time. The resulting aggregate demand densities may get larger than one over some intervals of time, as the individual demand densities from which they originate have not been checked for consistency with respect to the capacity constraints. High demand for a resource over some time interval indicates a critical resource/time interval pair, which requires prompt attention from the scheduler. This is the basis to the variable-ordering heuristic presented in this paper.

More precise probabilities may be obtained by iterating the propagation process. One way to do so consists in computing for each possible activity reservation the probability that this reservation will be available and that no better reservation will be available. The availability probability of a reservation can be approximated by the probability that the reservation does not violate any activity precedence constraint or capacity constraint (see section 5 for details). These probabilities can then be combined into new a priori start time and resource probability distributions, and the propagation process can be carried out all over again. The experimental results that we report in this paper have all been obtained without iterating the propagation process. We are planning to perform similar experiments with probability distributions obtained after iterating the propagation a variable number of times.

Notations

In the remainder of the paper the following notations will be used:

- $P^{PRIOR}(st_k=t)$ will denote the a priori probability that A_k will be scheduled to start at time t ,
- $P^{POST}(st_k=t)$ will be the a posteriori probability that A_k starts at time t , i.e. after accounting for activity precedence constraints,
- $P_N^{POST}(st_k=t)$ represents the same probability distribution after it has been normalized to express that A_k will start exactly once,
- $D_{kij}(t)$ represents A_k 's individual demand for R_{kij} at time t , and
- $D_{R_{kij}}^{agg}(t)$ will denote the aggregate demand for R_{kij} at time t .

3.4 ARR: A Variable Ordering Heuristic Based on Activity Resource Reliance

ARR, the variable ordering heuristic that we study in this paper, consists in looking for the resource/time interval pair that is the most contended for and the activity that relies most on the possession of that resource over that time interval. This activity is selected as the most critical one and hence is the next one to be scheduled.

The intuition behind this heuristic is the following. If activities that critically rely on the possession of highly contended resources were not scheduled first, it is very likely that, by the time the scheduler would turn its attention to them, the reservations that are the most appropriate for these activities would no longer be available.

The aggregate demand densities introduced in subsection 3.2 are used to identify the most demanded resource/time-interval pair. The activity that contributes most to the demand for the resource over the time interval (i.e. the activity with the largest individual demand for the resource over the time interval) is interpreted as the one that relies most on the possession of that

resource. Indeed the total demand of an activity A_k for one of its resource requirement R_{ki} is equal to A_k 's duration and is distributed over the different alternatives, R_{kij} , for that resource, and over the different possible times when A_k can be carried out. Consequently activities with a lot of slack or several resource alternatives tend to have fairly small individual demand densities at any moment in time. They rely less on the possession of a resource at any moment in time than activities with less slack and/or fewer resource alternatives. This allows ARR to account not only for inter-order interactions but also for intra-order interactions.

The advantage of this approach lies in its relative simplicity: look for the most critical resource/time-interval pair and select the activity that relies most on it. One may however contend that this heuristic does not consider slack as an independent component to activity criticality. Instead slack is only accounted for via resource contention. Another possible problem with this heuristic is that it only considers resource reliance with respect to the most contended resource. In general an activity A_k may require several resources R_{ki} . Rigorously A_k 's criticality should therefore account for each of these resources. It should account for the contention for each of the possible alternatives R_{kij} for these resources R_{ki} , and the reliance of A_k on the possession of each of these alternatives R_{kij} . Computation of such a criticality measure is likely however to be more expensive.

3.5 Three Value Ordering Heuristics

In the experiments that we ran, we considered the following three value-ordering heuristics:

3.5.1 LCV: A Least Constraining Value Ordering Heuristic

Least constraining value ordering heuristics are known for being very good at reducing search [Haralick 80, Dechter 88]. Similar heuristics have also been proposed for scheduling, even when viewed as an optimization problem. [Keng88], for instance, suggests the use of a least constraining value ordering heuristic as a primary criterion for selecting a reservation for an activity. The quality of the reservations is only used as a secondary criterion when there are several least constraining reservations to choose from. A similar heuristic is also outlined in [Muscettola 87]. The extremely small number of feasible solutions to a scheduling problem compared to the total number of schedules that one can possibly generate is what has made least constraining value ordering heuristics so attractive.

LCV is a least constraining value ordering heuristic where every reservation $\{(st_k=t, R_{k1j_1}, R_{k2j_2}, \dots, R_{kp_kj_{p_k}})\}$, for an activity A_k , is rated according to the probability $RESERV-AVAIL(st_k=t, R_{k1j_1}, \dots, R_{kp_kj_{p_k}})$ that it would not conflict with another activity's reservation, if one were to first schedule all the other remaining activities. The reservation with the largest such probability is interpreted as the least constraining one.

In our model, we express $RESERV-AVAIL(st_k=t, R_{k1j_1}, \dots, R_{kp_kj_{p_k}})$ as the product of the probability that $st_k=t$ will not result in the violation of an activity precedence constraint and the conditional probability that each resource $R_{k1j_1}, R_{k2j_2}, \dots, R_{kp_kj_{p_k}}$ will be available between t and $t + du_k$, given that $st_k=t$ does not result in the violation of an activity precedence constraint :

$$\begin{aligned}
& \text{RESERV-AVAIL}(st_k=t, R_{k1j_1}, \dots, R_{kp_kj_{p_k}}) \\
&= \frac{p\text{POST}(st_k=t)}{p\text{PRIOR}(st_k=t)} \times \prod_{R_{kij} \in \{R_{k1j_1}, \dots, R_{kp_kj_{p_k}}\}} \text{RESOURCE-AVAIL}(R_{kij}, t, t+du_k)
\end{aligned}$$

where $\text{RESOURCE-AVAIL}(R_{kij}, t, t+du_k)$ is the conditional probability that R_{kij} will be available between t and $t+du_k$, given that $st_k=t$ does not result in the violation of an activity precedence constraint.

We will approximate the (conditional) probability that R_{kij} will be available at some time τ for activity A_k with $\frac{D_{kij}(\tau)}{D_{R_{kij}}^{\text{aggr}}(\tau)}$. When approximating $\text{RESOURCE-AVAIL}(R_{kij}, t, t+du_k)$, one has to be careful not to come up with too pessimistic an estimate. Indeed it is tempting to assume that the (conditional) probability that R_{kij} will be available for A_k between t and $t+du_k$ is given by the product of $\frac{D_{kij}(\tau)}{D_{R_{kij}}^{\text{aggr}}(\tau)}$ over all possible start times τ between t and $t+du_k$. Depending on whether time is discrete or not, this product would be finite or infinite. In either case the approximation would be too pessimistic. Indeed this would be tantamount to supposing that the activities that contribute to $D_{R_{kij}}^{\text{aggr}}(\tau)$ have infinitely small durations, i.e. that these activities can possibly require R_{kij} at time τ without requiring it at $\tau-\delta\tau$ or $\tau+\delta\tau$. Instead, in order to account for the duration of these activities, we will assume that each resource R_{kij} is subdivided into a sequence of buckets of duration $\text{AVG}(du)$, where $\text{AVG}(du)$ is the average duration of the activities competing for R_{kij} . Consequently $\text{RESOURCE-AVAIL}(R_{kij}, t, t+du_k)$ is given by the probability that A_k can secure a number of buckets equal to its duration, i.e.:

$$\text{RESOURCE-AVAIL}(R_{kij}, t, t+du_k) = \left\{ \text{AVG}\left(\frac{D_{kij}(\tau)}{D_{R_{kij}}^{\text{aggr}}(\tau)}\right) \right\}^{\frac{du_k}{\text{AVG}(du)}}$$

where $\text{AVG}\left(\frac{D_{kij}(\tau)}{D_{R_{kij}}^{\text{aggr}}(\tau)}\right)$ is simply the average of $\frac{D_{kij}(\tau)}{D_{R_{kij}}^{\text{aggr}}(\tau)}$ taken between t and $t+du_k$.

3.5.2 HC: A Hill-Climbing Value Ordering Heuristic

The second value ordering heuristic that we tested simply consists in ranking an activity's possible reservations according to their utilities, i.e. preferences. Reservations with the highest preferences are the first ones to be tried.

3.5.3 INT: An Intermediate Value Ordering Heuristic

Our third value ordering heuristic combines features from the previous two. Each reservation is rated according to the probability that it would be available and that no better reservation would be available, if one were to first schedule all the other remaining activities.

3.6 Preliminary Experimental Results

A testbed was implemented that allows for experimentation with a variety of variable and value ordering heuristics based on the probabilistic framework described in subsection 3.2. The system is implemented in Knowledge Craft running on top of Common Lisp, and can be run either on a MICROVAX 3200 or on a VAX 8800 under VMS.

We performed some preliminary experiments to evaluate the four heuristics presented in this paper. These experiments were run on a set of 38 scheduling problems, involving between 3 and 5 orders and a total number of activities ranging between 10 and 20. The problems involved 3 or 4 resources. They involved only activities with a unique resource requirement (R_{k1}), for which there was one or several alternatives (R_{k1j}). Problems with both equally preferred and non equally preferred resource alternatives were included. The scheduling problems were built to reflect a variety of demand profiles: localized bottlenecks at the beginning, middle, and end of the problem span, global bottlenecks spanning the whole duration of the scheduling problems, and auxiliary bottlenecks were all included. Three different types of start time utility functions were allowed: all start times (between the earliest and latest start times) are equally preferred, late start times are preferred, and triangular start utility functions with a peak corresponding to the due date (minus the duration of the activity). Triangular utility functions were only assigned to the last activities of some orders. Time was discretized and a granularity equal to the third of the smallest activity duration was used. A discrete version of the formulas presented in this paper was used to compute the necessary probability distributions. The probabilities were computed using biased a priori probability distributions obtained by normalizing the utility functions over the domain of possible values of each variable. The granularity of the time intervals used for the ARR variable ordering heuristic varied from one resource to the other and was selected to be equal to the duration of the shortest activity requiring the resource.

Preliminary Experimental Results					
	RAND &HC	RAND &LCV	ARR &HC	ARR &LCV	ARR &INT
Search Efficiency	< 0.47 (> 0.27)	1.00 (0.00)	0.96 (0.05)	1.00 (0.00)	1.00 (0.00)
Schedule Value	not available	0.52 (0.08)	0.68 (0.06)	0.54 (0.06)	0.64 (0.05)

Figure 3-1: Average search efficiencies and schedule values for 5 combinations of variable and value ordering heuristics run on a preliminary set of 38 scheduling problems. The standard deviations appear between parentheses.

The experiments were measured along two dimensions: *search efficiency* (i.e. number of

operations to schedule over number of search states generated) and *global utility* of the solution as defined by a normalized objective function. The normalized objective functions were built so that the best possible schedules that could be built without checking for constraint violation would have a global value of 1. There was no guarantee however that a *feasible* schedule with global value of 1 could be built. In the ideal case the search would be performed without backtracking, and the number of search states generated would be equal to the number of activities to schedule (i.e. efficiency of 1). The quality of the schedules is more difficult to assert as the value of the objective function for the optimal schedule varied from one problem to the other and was in most cases smaller than 1. For this reason the values of the schedules should not be viewed as absolute measures. Instead they should only be used to compare the relative performances of the combinations of heuristics that we tried.

The table in Figure 3-1 reports the average search efficiencies and schedule values obtained with five combinations of variable and value ordering heuristics (for a total of 190 experiments). Standard deviations are provided between parentheses. RAND denotes a random variable ordering heuristic, where the next activity to be scheduled is selected at random from the remaining unscheduled activities. Search was stopped when it would require more than 50 search states. For RAND&HC, this cutoff rule had to be used in 12 of the 38 experiments. It did not have to be used for any of the other heuristics. The average search efficiency of RAND&HC is therefore even worse than 0.47. Because search did not terminate in almost a third of the runs with RAND&HC, no good estimate of the value of the schedules produced by this heuristic could be obtained.

3.7 Discussion

The results reported in Figure 3-1 clearly indicate the importance of a good variable ordering heuristic to increase search efficiency (e.g. ARR&HC vs. RAND&HC). They also indicate that a least constraining value ordering heuristic can make up for a poor variable ordering heuristic (e.g. RAND&HC vs. RAND&LCV and RAND&LCV vs. ARR&LCV). In the examples that we ran the ARR variable ordering heuristic and the LCV value ordering heuristics both contributed to limit search. The quality of the schedules produced by LCV is however very poor when compared to the other two value ordering heuristics (ARR&HC or ARR&INT vs. ARR&LCV). In particular ARR&INT performed as well as ARR&LCV as far as the efficiency of the search is concerned (neither of them had to backtrack in any of the 38 examples) but produced much better schedules. Even a simple value ordering heuristic like HC resulted in very little amount of backtracking when coupled with our variable ordering heuristic (see ARR&HC). Overall HC produced the best schedules although it required more search than INT and LCV, when coupled with ARR. There seem therefore to be a tradeoff between the amount of search performed and the quality of the resulting solution. If little time is available to come up with a solution the most promising values may be the least constraining ones as they are the least likely to result in backtracking. On the other hand, when there is more time available, one may consider looking at riskier values if they are likely to produce better solutions. A value ordering heuristic could accordingly be designed that accounts for the time available to find a schedule.

3.8 Concluding Remarks

In this chapter, an activity-based approach to scheduling has been investigated. Because of its greater flexibility, such an approach is expected to allow for the construction of better schedules than approaches using order-based or resource-based scheduling or even combinations of the two. The price to pay for this flexibility is the potential overhead involved in the selection of the next decision point on which to focus attention. While order-based and resource-based scheduling typically require only the computation of criticality measures for each order or resource in the system, an activity-based scheduling approach may potentially require the computation of similar measures for each of the activities to be scheduled. In the simplest scenario, these measures need moreover to be recomputed every time a new activity has been scheduled. It is therefore important that the computation of these measures can be performed at a relatively cheap computational cost. One may also consider scenarios where several activities are scheduled before new criticality measures are computed.

We have presented a probabilistic framework that successively accounts for both intra-order and inter-order interactions. Although such a two step propagation involves a slight loss of precision in the way it accounts for interactions, it has the advantage of having a relatively low computational cost [Sadeh88]. More accurate probability distributions may always be obtained by iterating the propagation process. Our probabilistic framework allows for the definition of a variety of variable and value ordering heuristics. In this paper, we have studied a simple variable-ordering heuristic, ARR, that looks for the most contended resource/time interval pair and the activity that relies the most on the possession of that resource/time interval pair. Preliminary experiments with the heuristic indicate that it greatly contributes to increasing search efficiency. Additionally our experiments with three value ordering heuristics seem to indicate that least constraining value ordering heuristics such as the one advocated in [Keng88] may not be the only viable way to maintain search at an acceptable level. Instead, in our experiments, other value ordering heuristics, when coupled with our variable ordering heuristic, produced much better schedules without significantly increasing search.

Our variable ordering heuristic is not perfect. For instance it measures activity criticality only with respect to one resource (the most contended resource/time interval pair). While the heuristic performed well in problems where each activity requires only one resource (for which there may or may not be alternatives), it may not be as effective for activities requiring several resources. We are currently looking at other variable ordering and value ordering heuristics. We are also pursuing our experiments with the heuristics presented in this paper. In particular we still have to study the behavior of these heuristics on larger scheduling problems (i.e. more than 100 activities).

Our long term interest is in the identification of a set of (texture) measures characterizing the search space, that can be used to both structure and guide search in that space. Measures of variable criticality (variable ordering heuristics) and estimates of value goodness (value ordering heuristics) are examples of such measures [Fox89].

CHAPTER 4: Representation

4.1 Introduction

In this chapter we present a framework for representing knowledge needed to perform constraint-directed reasoning with special emphasis on the domain of scheduling. In addition, this framework serves as the foundation of the implementation-level representation that we have adopted in the project. The representation is expressed in the CRL language [KC 86]. A scheduler that uses the representation presented in this report is described in chapter 3.

The main conceptual primitives in our work are *activities*, *resources*, *production units*, *states*, and *constraints*. These primitives provide an extensible framework that can be used to represent the relevant aspects of particular problem solving environments where constraint-directed reasoning is used. In addition, these primitives are represented at various levels of conceptual abstraction depending on the granularity of knowledge. For example, an *operation* is a specialization of an activity. An important component of the representational framework is the *relations* that connect the primitives and their instantiations. The main types of relations are temporal and causal. In general, we distinguish between prototype descriptions and their instantiations [KC 86] into specific manifestations of the corresponding concepts. The concepts presented in this section are represented in the figures as *schemata*. A schema encapsulates information and has an identifying name. A schema has a set of slots that describe attributes of the schema. *Meta* information can be attached to a slot, a slot value or schema. Meta information is information *about* the slot, slot value, or schema and is independent of the meaning represented by these entities. In presenting schemata, we use the following convention: schemata are shown in **boldface**, slots are shown in small CAPITALS and meta-slots are shown in *italics*. The range restrictions are as described in [KC 86].

4.2 Activities

Activities are the subject of scheduling decisions. The specification of activities includes the temporal and causal relations that connect them as well as their organization as *aggregate activities* into larger constructs. An activity is *elaborated* into an aggregate activity (an activity network) whose activities are *part-of* the aggregate activity. For example, a milling-operation has an elaboration *milling-operation-network*, which in turn has two activities, *milling-setup* and *milling-run*. The primitive relation *part-of* connects the activity network to its component activities. Thus, the *elaboration-of* relation separates an activity from its detailed description and facilitates the representation of multiple elaborations of the same activity at different levels of abstraction. Figure 4-1 depicts the pair of primitive relations *elaboration-of* and *has-elaboration*.

```
{ {elaboration-of
  IS-A: relation
  INVERSE: has-elaboration
  DOMAIN: (or (type is-a state) (type is-a activity))
  RANGE: (or (type is-a state) (type is-a activity))
  TRANSITIVITY: (step elaboration-of t)}}
```

```
{ {has-elaboration
  IS-A: relation
  INVERSE: elaboration-of
  DOMAIN: (or (type is-a state) (type is-a activity))
  RANGE: (or (type is-a state) (type is-a activity))
  TRANSITIVITY: (step has-elaboration t) }}
```

Figure 4-1: Relations representing elaboration of activities

In the manufacturing environment, activities are typically of two types: (a) operations associated with the manufacturing processes to produce a part/product, and (b) supporting activities, such as maintenance and repair. Temporal and causal relations organize operations into production plans that indicate which operations need to succeed each other, which ones can be executed in parallel and which resources each operation needs for its execution. The pair of primitive relations *has-next-activity* and *next-activity-of* shown in 4-2 links an activity with its successor(s) and predecessor(s).

```
{ {has-next-activity
  IS-A: relation
  INVERSE: next-activity-of
  DOMAIN: (type is-a activity)
  RANGE: (type is-a activity) }}
```

```
{ {next-activity-of
  IS-A: relation
  INVERSE: has-next-activity
  DOMAIN: (type is-a activity)
  RANGE: (type is-a activity) }}
```

Figure 4-2: Precedence relations between activities

Another way to aggregate operations is in terms of an *order* which indicates how many parts need to be produced for the order's fulfillment. An operation is the basic unit of action in a scheduling environment. It defines a transformation of the world from one state to another so that at the end a part (an order) is produced.

Figures 4-3 and 4-4 presents an aggregate activity and a process plan which is one kind of aggregate activity. Each aggregate activity has a type that is either a conjunctive or disjunctive abstraction. A conjunctive abstraction indicates that the aggregate activity can be expressed as a sequence of activities at the next lower level of the abstraction hierarchy. A disjunctive abstraction indicates that the aggregate activity can be expressed as a set of alternative activities at the next lower level of the abstraction hierarchy.

```
{ {aggregate-activity
  IS-A: activity
  TYPE:
    (Range (or "and" "or"))
  ELABORATION-OF:
  HAS-SUBACTIVITY: }}
```

Figure 4-3: Aggregate Activity

```
{ {process-plan
  IS-A: aggregate-activity }}
```

Figure 4-4: A Process Plan

The primitive relation *has-subactivity*, presented in figure 4-5, denotes the set of activities that comprise the prototype aggregate activity. The primitive relation *subactivity-of*, presented in figure 4-6, relates a subactivity back to the aggregate activity to which it belongs. These relations are used to describe process abstractions.

```
{ {has-subactivity
  IS-A: relation
  INVERSE: subactivity-of
  DOMAIN: (type is-a aggregate-activity)
  RANGE: (type is-a activity)
  TRANSITIVITY: (repeat (step has-subactivity t) 1 inf) }}
```

Figure 4-5: The relation has-subactivity

```
{ {subactivity-of
  IS-A: relation
  INVERSE: has-subactivity
  DOMAIN: (type is-a activity)
  RANGE: (type is-a aggregate-activity)
  TRANSITIVITY: (repeat (step subactivity-of t) 1 inf) }}
```

Figure 4-6: The relation subactivity-of

In order to be performed, activities require one or more resources. For example, in order to perform a *cutting operation*, the required resources are a *machine* on whose machine head a *cutting tool* can be affixed, the *part* on which the cutting operation is to be performed, a *fixture* that immobilizes the part on the machine's bed, and an *operator* to operate the machine and to load and unload the part and fixture. The resource requirements of activities are expressed by the primitive relations *required-resource*, presented in figure 4-7 and *resource-for*, presented in figure 4-8. The representation of resources is described in section 4. In our model, it is assumed that all resources required by an activity, are required for the whole duration of the activity. An additional assumption that we make is that 100% of each resource is required for the performance of the associated activity. This is a simplifying assumption that might not be true in actual manufacturing environments. For example, depending on the duration of a particular cutting operation, an operator might be able to operate (e.g., load another part, fixture it and start an operation) another machine during the cutting operation.

```
{ {required-resource
  IS-A: relation
  DOMAIN: (type is-a activity)
  RANGE: (type is-a resource)
  INVERSE: resource-for }}
```

Figure 4-7: The relation required-resource

```
{ {resource-for
  IS-A: relation
  DOMAIN: (type is-a resource)
  RANGE: (type is-a activity)
  INVERSE: required-resource }}
```

Figure 4-8: The relation resource-for

Assembling the primitives already defined and discussed for describing manufacturing activities, we now proceed to put together the representation of an activity, shown in figure 4-9 and its specialization of interest, an *operation*, shown in figure 4-10. The manufacturing unit which is transformed by the operation is expressed through the attribute *operates-on*.

```
{{activity
  IS-A: concept
  HAS-TIME-INTERVAL:
  ENABLED-BY:
  CAUSE:
  ELABORATION-OF:
  HAS-ELABORATION:
  HAS-SUBACTIVITY:
  SUBACTIVITY-OF:
  COST:
  SCHEDULING-STATUS: }}
```

Figure 4-9: Representation of an activity

```
{{operation
  IS-A: activity
  OPERATION-NUMBER:
  REQUIRES:
  RUN-TIME:
  LOAD-TIME:
  UNLOAD-TIME:
  OPERATES-ON:
  HAS-NEXT-OPERATION:
  NEXT-OPERATION-OF: }}
```

Figure 4-10: Representation of an operation

4.3 States

A state is the collection of conditions under which an activity can be performed, or the collection of new conditions produced by the activity. An activity is connected to its precondition and postcondition states by *causal relations*. The primitive pair of inverse relations *enables* and *enabled-by*, shown in figure 4-11, specify the connection between an activity and its enabling state.

```
{ {enabled-by
  IS-A: causal-relation
  INVERSE: enable
  DOMAIN: (type is-a activity)
  RANGE: (type is-a state)
  TRANSITIVITY: (step enabled-by t) }}
```

```
{ {enable
  IS-A: causal-relation
  INVERSE: enabled-by
  DOMAIN: (type is-a state)
  RANGE: (type is-a activity)
  TRANSITIVITY: (step enable t) }}
```

Figure 4-11: Relations defining enablement of activities

In a similar fashion, the pair of inverse primitive relations, depicted in figure 4-12, *cause* and *caused-by* defines the connection between an activity and its resulting set of conditions.

```
{ {cause
  IS-A: causal-relation
  INVERSE: caused-by
  DOMAIN: (type is-a activity)
  RANGE: (type is-a state)
  TRANSITIVITY: (step cause t) }}
```

```
{ {caused-by
  IS-A: causal-relation
  INVERSE: cause
  DOMAIN: (type is-a state)
  RANGE: (type is-a activity)
  TRANSITIVITY: (step caused-by t) }}
```

Figure 4-12: Relations defining activity postconditions

The abstraction of state information is performed using the same operators (conjunction and disjunction) as for activity information, resulting in *aggregate states* that are related to their component states via the *has-sub-state* relation. In turn, the component states are related back to the aggregate state via the *sub-state-of* relation. An aggregate state that is a disjunct is true if any of its sub-states is true. An aggregate state that is a conjunct should have all its sub-states true in order to be true. Figure 4-13 shows the representation of the relations that define state

aggregation.

```
{ {substate-of
  IS-A: relation
  INVERSE: has-substate
  DOMAIN: (type is-a state)
  RANGE: (type is-a aggregate-state)
  TRANSITIVITY: (step substate-of t) }}
```

```
{ {has-substate
  IS-A: relation
  INVERSE: substate-of
  DOMAIN: (type is-a aggregate-state)
  RANGE: (type is-a state)
  TRANSITIVITY: (step has-substate t) }}
```

Fig. 4-13: Relations defining state aggregation

States, as well as activities, persist for particular time intervals. The relation *has-time-interval*, shown in figure 4-14, is associated with each state and activity in a scheduling system. For a detailed representation of temporal relations, see section 6.2.

```
{ {has-time-interval
  IS-A: relation
  DOMAIN: (or (type is-a activity) (type is-a state)) }}
```

Figure 4-14: Representation of the *has-time-interval* relation

The following figures, figure 4-15 and 4-16, represent a prototype *state* and *aggregate-state*.

```
{{state
  IS-A: concept
  HAS-TIME-INTERVAL:
  ENABLE:
  CAUSED-BY:
  ELABORATION-OF:
  HAS-ELABORATION:
  SUBSTATE-OF:
  SCHEDULING-STATUS: }}
```

Figure 4-15: Representation of a state

```
{{aggregate-state
  IS-A: state
  HAS-SUBSTATE: }}
```

Figure 4-16: Representation of an aggregate state

4.4 Resources

In this section, we present the hierarchical representation of resources to enable reasoning at different levels of precision in allocating a resource. In the centralized system, we do not consider functional resource groupings corresponding to work areas (resource groupings that support particular production processes and/or process steps), cells (groupings of identical machine types where tradeoffs based on particular machine characteristics can be factored into allocation decisions) and individual machines. The aggregate resource abstractions (e.g, work area and cell) will be considered as different agents in the distributed scheduling system. Considering different aggregate resources as agents necessitates the definition of relations of *ownership* of resources that are part of the aggregation, and relations *has-resource* and *is-resource-of* to define the mutually exclusive groupings of resources belonging to an aggregate resource.

Viewing resources at various aggregate levels impacts the level of granularity of capacity definitions. In general, the capacity of a resource is the number of items that the resource can process simultaneously. The term "item" is a general term and can refer to different units on which the resource is operating. For example, a machine operator has capacity 1 and (usually) operates one machine; a milling machine can be configured to cut 3 identical parts at the same time (capacity 3).

We differentiate resources into groupings of various types. The two basic groupings that we distinguish are *stationary* and *mobile* resources. Stationary resources are the ones that have a fixed location. In a manufacturing organization, examples of stationary resources are machines

and workstations. Examples of mobile resources are human operators and fixtures that can be loaded on different machines. The *current location* attribute holds location information for movable resources. In some situations, a stationary resource can have moving parts. For example, a milling machine can have a moving head that can take more than one position. Hence, in the representation of such a resource, the *current position* of the machine head has to be noted. Associated with fixtures are *load* and *unload* operations that require the fixture and the position on which it is to be loaded as resources. The *status* of a resource indicates whether it is available or not.

The next three figures, figure 4-17, 4-18 and 4-19, represent an abstract resource, and its specializations of a milling machine (a stationary resource) and a fixture (a mobile resource).

```
{{resource
  ISA-A: physical-object
  HAS-RESOURCES:
  RESOURCE-OF:
  CAPACITY:
  OWNED-BY: }}
```

Figure 4-17: The representation of a resource

```
{{milling-machine
  IS-A: resource
  HAS-WORK-BED: milling-bed
  CURRENT-POSITION:
    range: (TYPE instance bed-position)
  STATUS: }}
```

Figure 4-18: The representation of a milling machine

```
{{fixture
  IS-A: resource
  FIXTURE-NUMBER:
  CURRENT-LOCATION:
  STATUS: }}
```

Figure 4-19: The representation of a fixture

4.5 Production Units

Production units are the entities which are transformed by operations during the manufacturing process. In this work, we are interested in modeling production units from the standpoint of scheduling. This perspective, obviously constrains representational completeness, since modeling considerations such as those related to design, process planning and marketing are not included. The central concept in modeling production units is the *part*. The manufacture of a part enters the manufacturing system through a **work-order**, the representation of which is shown in figure 4-20, that specifies the QUANTITY of the part ordered, the DUE-DATE of the order, the SERIAL-NUMBER and the order's PRIORITY.

```
{{work-order
  IS-A: concept
  SERIAL-NUMBER:
  FOR-PART:
  QUANTITY:
  PRIORITY:
  DUE-DATE: }}
```

Figure 4-20: Representation of a work-order

One central characteristic of a part is the production process through which it is manufactured. An abstracted representation of the process is the *process-plan* that has been discussed in section 2. Production plans are usually prototypical in that at the time of their construction they have no information (neither can they anticipate) factory floor configurations and scheduling constraints.²⁹ The scheduler instantiates the prototypical process plan for filling a work order to reflect the realities of the factory floor at scheduling time. Another characteristic of a part is the required material that is used to produce the part. This is related to inventory concerns of availability of material. Currently, we assume that the required material is present when needed for the production of the part. Figure 4-21 shows the representation of a part.

```
{{part
  IS-A: physical-object
  WORK-ORDER:
  PROCESS-PLAN:
  PRODUCTION-QUANTITY:
  PART-ID: }}
```

Figure 4-21: Representation of a part

²⁹In this year's effort on the project, we have started to address the issues in the interactions of a process planner with a scheduler in order to produce process plans that optimize scheduling concerns.

4.6 Constraints

In general, there are five types of constraints that a scheduler should take into consideration.

- Physical constraints. Physical constraints include, number of machines, fixtures, setup and run times for each operation.
- Organizational constraints. Examples of organizational constraints include meeting due dates, reducing Work in Process, increase machine utilization, and enhance throughput.
- Preferential constraints. Examples of preferential constraints include preference for using a particular machine for an operation (perhaps because of its speed or accuracy), or using a particular human operator (perhaps because of his skill).
- Enablement constraints. These refer to constraints, the fulfillment of which creates a state that enables the execution of an activity. For example, a process plan embodies enablement constraints.
- Availability constraints. These constraints refer to the availability of particular resources at scheduling time. For example, a machine may become unavailable because of breakdown, the assignment of a third shift makes extra resources available for scheduling.

4.6.1 Representing Constraints

The constraints found in scheduling mainly restrict (a) the choice of value(s) for some variable (e.g., the due date of an order) or (b) the relation between two or more variables (e.g., the next operation to be performed). We differentiate two types of variables: simple variables and aggregate variables. Simple variables are represented as slots in a CRL schema [KC 86]; aggregate variables are CRL schemata. For example, a *time interval* during which an activity can be potentially scheduled, is an aggregate variable whose slots include the simple variables *start-time* of the activity, *end-time* of the activity, and activity *duration*. In the case of a simple variable, constraint-related knowledge is stored in a meta-slot. In the case of an aggregate variable, it is stored in a meta-schema. In the formalization of the scheduling problem (see chapter 2 of this document), the variables of interest are the activity start times, the resources required by each activity, and the duration of each activity.

The pair of primitive relations *constrains/constrained-by*, shown in figure 4-22, links a simple variable with the constraints that affect its values.

```
{ {constrains
  IS-A: relation
  INVERSE: constrained-by
  DOMAIN: (type is-a constraint)
  RANGE: (type is-a simple-variable) }}
```

```
{ {constrained-by
  IS-A: relation
  INVERSE CONSTRAINS:
  DOMAIN: (type is-a simple-variable)
  RANGE: (type is-a constraint) }}
```

Figure 4-22: Relations associating a single constraint to a variable

The pair of primitive relations *simple-variable-affects/affected-by-simple-variable*, shown in figure 4-23 links a simple variable to the aggregate variable it affects.

```
{ {affected-by-simple-var
  IS-A: relation
  INVERSE SIMPLE-VAR-AFFECTS:
  DOMAIN: (type is-a aggregate-variable)
  RANGE: (type is-a simple-variable) }}
```

```
{ {simple-var-affects
  IS-A: relation
  INVERSE AFFECTED-BY-SIMPLE-VAR:
  RANGE: (type is-a aggregate-variable)
  DOMAIN: (type is-a simple-variable) }}
```

Figure 4-23: Relations linking simple and aggregate variables

Figure 4-24 depicts a meta-slot prototype for simple variables.

```
{ {variable
  IS-A: concept } }
```

```
{ {simple-variable
  IS-A: variable
  CONSTRAINED-BY:
  SIMPLE-VAR-AFFECTS:
  UTILITY-VALUE:
  IMPORTANCE:
  OF-SCHEMA:
  OF-SLOT: } }
```

Figure 4-24: Representation of a simple variable

The slot *of-schema* indicates the schema with the slot to which the meta-slot is attached. The slot *of-slot* indicates the slot in the schema to which the meta-slot is attached.

We have seen the representation of the constraining relations between simple and aggregate variables. In turn, aggregate variables can affect the values of higher level aggregate variables to which they can be linked. This recursive influence is expressed by the pair of primitive relations *affected-by-aggr-var/aggr-variable-affects*, shown in figure 4-25.

```
{ {affected-by-aggr-var
  IS-A: relation
  INVERSE AGGR-VAR-AFFECTS:
  DOMAIN: (type is-a aggregate-variable)
  RANGE: (type is-a aggregate-variable) } }
```

```
{ {aggr-var-affects
  IS-A: relation
  INVERSE: affected-by-aggr-var
  RANGE: (type is-a aggregate-variable)
  DOMAIN: (type is-a aggregate-variable) } }
```

Figure 4-25: Relations connecting aggregate variables

Figure 4-26 depicts the meta-schema prototype representation of an aggregate variable.

```
{{aggregate-variable
  IS-A: variable
  AGGR-VAR-AFFECTS:
  AFFECTED-BY-AGGR-VAR:
  AFFECTED-BY-SIMPLE-VAR:
  UTILITY-VALUE:
  IMPORTANCE: }}
```

Figure 4-26: Representation of an aggregate variable

In the model, we treat explicitly two types of constraints, *required constraints* and *preferential constraints* [Fox 83c]. The degree of satisfaction of a preferential constraint is expressed by a *utility function* ranging between 0 and 1. A value of 0 utility is non-admissible; a value of 1 is optimal. Variables, simple as well as aggregate, can be constrained by more than one constraint. So, utility values are not only associated with constraints but also with simple and aggregate variables. The utility value associated with a simple variable (slot) is calculated by taking the weighted sum (with constraint importance as the weight) of the utilities of all the constraints that affect the slot's value. The utility value associated with an aggregate variable (schema) is calculated by taking the weighted sum of the utilities of all the constraints that affect the aggregate variable.

Constraints differ in *importance*. A particular constraint could have different importance depending on the context in which it is applied. The importance of a constraint is specified by a value between 0 and 1. An importance of 0 implies that the constraint should not be considered, and 1 signifies maximum importance. The actual level of importance is relative to the importance of the other constraints under consideration. The "importance" slot contains the value of a constraint's importance. The measure of importance of a constraint may be viewed as a weight that can be combined with a constraint's utility value to form a weighted combination of utilities. Constraints also differ in *relevance*. Depending on the context, a constraint may be more relevant than others.

During the construction of a schedule, it may be found that one or more constraints may not be satisfiable. For example, an operation cannot be scheduled on a resource at a particular time because the resource is unavailable. The system should look for alternative ways to satisfy the scheduling goal. The selection of an alternative to a specified constraint is called a *constraint relaxation*. Constraint relaxations are specified using the *relaxation-spec* relation, shown in figure 4-27. Moreover, relaxation specifications can be either *continuous* or *discrete* [Fox 83c], shown in figures 4-28 and 4-29. A discrete relaxation specifies a number of discrete alternatives (e.g., different machine alternatives), whereas a continuous one specifies relaxations over a continuous variable such as time (e.g., due date can be continuously relaxed).

In general, it may not be possible to find a schedule that satisfies all specified constraints. The scheduler must have a way of choosing the best among possible relaxations. The means by which a scheduler makes such a decision is a system of utilities. The particular utility associated with each relaxation fills the utility slot in the specializations of the relaxation-spec schema. A

constraint may have one of two effects on a schedule: it may determine the *admissibility* of a schedule, or it may determine the *acceptability* of a schedule. Admissibility determines the legality of a schedule against constraints that cannot be relaxed.

```
{ {relaxation-specification
  IS-A: concept }}
```

```
{ {relaxation-spec
  IS-A: relation
  DOMAIN: (type is-a constraint)
  RANGE: (type is-a relaxation-specification) }}
```

Figure 4-27: Representation of constraint relaxations

```
{ {continuous-relax-spec
  IS-A: relaxation-specification
  UTILITY-FUNCTION }}
```

Figure 4-28: Continuous constraint relaxation

```
{ {discrete-relax-spec
  IS-A: relaxation-specification
  UTILITY-FUNCTION: }}
```

Figure 4-29: Discrete constraint relaxation

The discrete type of constraint relaxation is further divided into two types, *exclusive or* relaxation where only one of the alternatives can be chosen and *inclusive or* where more than one alternative could be chosen. These are shown in figure 4-30.

```
{ {exor-relax-spec
  IS-A: discrete-relax-spec
  UTILITY-FUNCTION: } }
```

```
{ {inor-relax-spec
  IS-A: discrete-relax-spec
  UTILITY-FUNCTION: } }
```

Figure 4-30: Exclusive and inclusive OR relaxations

Constraints are associated with particular appropriate relaxations through the pair of relations *has-relaxation-spec* and *relaxation-spec-of*, shown in figure 4-31.

```
{ {has-relaxation-spec
  IS-A: relation
  INVERSE: relaxation-spec-of
  DOMAIN: (type is-a constraint)
  RANGE: (type is-a relaxation-spec)
  TRANSITIVITY: (step has-relaxation-spec t) } }
```

```
{ {relaxation-spec-of
  IS-A: relation
  INVERSE: has-relaxation-spec
  DOMAIN: (type is-a relaxation-spec)
  RANGE: (type is-a constraint)
  TRANSITIVITY: (step relaxation-spec t) } }
```

Figure 4-31: Representation of the relations
has-relaxation-spec/relaxation-spec-of

Having assembled the conceptual pieces that define a *constraint*, we present the constraint representation in figure 4-32.

```
{ {constraint
  IS-A: concept
  RELAXATION-SPEC:
  UTILITY-VALUE:
  RELEVANCE:
  IMPORTANCE:
  CONSTRAINS: }}
```

Figure 4-32: Representation of a constraint

4.6.2 Representing temporal relations

In our model, the relations among variables that we consider are primarily temporal relations. In particular, Allen's 13 temporal relations defined over time intervals [Allen 88] are represented in figures 4-33 and 4-34. The temporal relations express the precedence relations among manufacturing activities in the process plan. The constraints associated with the activities are checked for consistency and propagated over the temporal relations. For a detailed explanation of this process, see chapter 2 of this report. These relations are:

- *before*: specifies that an activity or state takes place before another activity or state in time. The inverse of *before* is the *after* relation.
- *meets*: specifies that an activity or state takes place before, but without any intervening time, another activity or state. Its inverse is *met-by*.
- *overlaps*: specifies that an activity or state begins before another activity or state in time, but ends after the second begins and before it ends. Its inverse is *overlapped-by*.
- *during*: specifies that an activity or state takes place during another activity or state in time. Its inverse is *contains*.
- *starts*: specifies that an activity or state begins at the same time as another activity or state (the two activities or states do not, however need to end at the same time). Its inverse is *started-by*.
- *finishes*: specifies that an activity or state finishes at the same time as another activity or state (the two activities, or states do not, however, need to start at the same time). Its inverse is *finished-by*.
- *equals*: specifies that an activity or state share the same time interval with another activity or state. It is a reflexive relation.

**{{time-relation
IS-A: relation }}**

**{{before
IS-A: time-relation
INVERSE: after }}**

**{{after
IS-A: time-relation
INVERSE: before }}**

**{{time-equal
IS-A: time-relation
INVERSE: time-equal }}**

**{{meets
IS-A: time-relation
INVERSE: met-by }}**

**{{met-by
IS-A: time-relation
INVERSE: meets }}**

Figure 4-33: Five of Allen's Temporal Relations

{{ overlaps
IS-A: **time-relation**
INVERSE: **overlapped-by** }}

{{ overlapped-by
IS-A: **time-relation**
INVERSE: **overlaps** }}

{{ during
IS-A: **time-relation**
INVERSE: **contains** }}

{{ contains
IS-A: **time-relatic.**
INVERSE: **during** }}

{{ starts
IS-A: **time-relation**
INVERSE: **started-by** }}

{{ started-by
IS-A: **time-relation**
INVERSE: **starts** }}

{{ finishes
IS-A: **time-relation**
INVERSE: **finished-by** }}

{{ finished-by
IS-A: **time-relation**
INVERSE: **finishes** }}

Figure 4-34: Representation of the rest of Allen's Temporal Relations

Allen's temporal relations are the attributes of a conceptual object, the *time object*, depicted in figure 4-35.

```
{{time-object
  IS-A: conceptual-object
  BEFORE:
  AFTER:
  TIME-EQUAL:
  MEETS:
  MET-BY:
  OVERLAPS:
  OVERLAPPED-BY:
  DURING:
  CONTAINS:
  STARTS:
  STARTED-BY:
  FINISHES:
  FINISHED-BY: }}
```

Figure 4-35: Representation of a time object

Whereas Allen's temporal relations can be used to express the occurrence of events that are *relative* to one another in a temporal sense, in scheduling there also arises the need to represent the occurrence of events in an *absolute* temporal sense. For example, the fact that operation1 has to precede operation2 can be expressed by "operation1 before operation2". On the other hand, the reservation of resource1 (required by operation1) for the time interval [t1, t2] is the expression of an event in an absolute temporal sense. Since, in scheduling, one is interested in the persistence of facts over time, we have chosen the *time interval* as the basic time primitive object. Characteristics of specific time points delineating a specific time interval (e.g., start and end times of an operation) are defined by associating each interval with a specific *time-line*. This association is made through the pair of inverse primitive relations *dates* and *dated-by*, shown in figure 4-36.

```
{ { dated-by
  IS-A: time-relation
  INVERSE: dates
  DOMAIN:
    (or (type is-a time-point) (type is-a time-interval))
  RANGE: (type is-a time-line) }
```

```
{ { dates
  IS-A: time-relation
  INVERSE: dated-by
  RANGE:
    (or (type is-a time-point) (type is-a time-interval))
  DOMAIN: (type is-a time-line) }
```

Figure 4-36: The relations *dates* and *dated-by*

The representation of a time-line, depicted in figure 4-37, makes provisions for units of time, a scale and functions to manipulate time. The slot *point-form* describes how a particular time unit is represented. For example, if we want to represent weekly time lines, time is represented as a pair of (year, week). The range of values for "year" are positive numbers, whereas the values for "week" range from 0 to 52. The *start-point* and *end-point* attributes indicate respectively the start and end point of the time-line. The attribute *granularity* provides an indication of the precision of the time line. *Add* and *subtract* store procedures for adding and subtracting time periods from each other.

```
{ { time-line
  IS-A: time-object
  POINT-FORM:
  START-POINT:
  END-POINT:
  GRANULARITY:
  ADD:
  SUBTRACT: }
```

Figure 4-37: Representation of a time line

In the definition of *time-interval*, shown in figure 4-38, attributes of importance are the various probability distributions that are propagated over the temporal relations among the activities so as to (a) account for the interactions induced by the problem constraints (intra-order, and inter-order interactions), and (b) to identify critical activities and promising resource reservations for these activities. For a detailed description of this process, refer to chapter 2.

```
{ {time-interval
  IS-A: time-object
  TIME-INTERVAL-OF:
  START-TIME:
  END-TIME:
  DURATION:
  APRIORI-S-T-DISTRIB:
  NORMALIZED-APRIORI-S-T-D:
  APRIORI-EARLIEST-START-TIME:
  APRIORI-LATEST-START-TIME:
  POSTERIOR-S-T-DISTRIB:
  NORMALIZED-POSTERIOR-S-T-D:
  DATED-BY: }}
```

Figure 4-38: Representation of a time interval

Since in this chapter, our concern is representation, we present the definition of the attributes that express probabilities. The *apriori-s-t-distrib* denotes the a priori probability that an activity A_k will be scheduled to start at time t , the *normalized-apriori-s-t-d*, is the normalized a priori start time distribution. *Apriori-earliest-start-time* and *apriori-latest-start-time* correspondingly refer to the a priori probability distributions that the earliest (latest) start time of activity A_k will be scheduled at time t . The *posterior-s-t-distrib* is the a posteriori probability that A_k will start at time t , i.e. after accounting for activity precedence constraints. Finally, the attribute *normalized-posterior-s-t-d* denotes the same probability distribution after it has been normalized to express that A_k will start exactly once.

References

- [Allen 83] J.F.Allen.
Maintaining Knowledge about Temporal Intervals.
Communications of the ACM 26(11):832-843, 1983.
- [Allen 84] J.F.Allen.
Towards a General Theory of Action and Time.
Artificial Intelligence 23(2):123-154, 1984.
- [Baker 74] K.R. Baker.
Introduction to Sequencing and Scheduling.
Wiley, 1974.
- [Bell 84] Colin E. Bell and Austin Tate.
Using Temporal Constraints to Restrict Search in a Planner.
Technical Report AIAI-TR-5, Artificial Intelligence Applications Institute,
University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, United
Kingdom, 1984.
- [Davis 87] Ernest Davis.
Constraint Propagation with Interval Labels.
Artificial Intelligence 32:281-331, 1987.
- [Dechter 88] Rina Dechter and Judea Pearl.
Network-Based Heuristics for Constraint Satisfaction Problems.
Artificial Intelligence 34(1):1-38, 1988.
- [Erman 80] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser and D. Raj Reddy.
The Hearsay-II Speech Understanding System: Integrating Knowledge to
Resolve Uncertainty.
Computing Surveys 12(2):213-253, June, 1980.
- [Fox 83] M. Fox.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
PhD thesis, Department of Computer Science, Carnegie-Mellon University,
1983.
- [Freuder 82] E.C. Freuder.
A Sufficient Condition for Backtrack-free Search.
Journal of the ACM 29(1):24-32, 1982.
- [Haralick 80] Robert M. Haralick and Gordon L. Elliott.
Increasing Tree Search Efficiency for Constraint Satisfaction Problems.
Artificial Intelligence 14(3):263-313, 1980.
- [Johnson 74] L.A. Johnson and D.C. Montgomery.
*Operations Research in Production Planning, Scheduling, and Inventory
Control*.
Wiley, 1974.
- [KC 86] *KnowledgeCraft Reference Manual*
Carnegie Group Inc., Pittsburgh, PA., 1986.

- [Laird, Newell & Rosenbloom 87] Laird, J.E., Newell, A., Rosenbloom, P.S.
SOAR: An Architecture for General Intelligence.
Artificial Intelligence 33(1):1-64, 1987.
- [Lepage 78] G. Peter Lepage.
A New Algorithm for Adaptive Multidimensional Integration.
Journal of Computational Physics 27:192-203, 1978.
- [LePape 87] Claude Le Pape and Stephen F. Smith.
Management of Temporal Constraints for Factory Scheduling.
Technical Report, The Robotics Institute, Carnegie Mellon University,
Pittsburgh, PA 15213, 1987.
also appeared in Proc. Working Conference on Temporal Aspects in
Information Systems, Sponsored by AFCET and IFIP Technical
Committee TC8, North Holland Publishers, Paris, France, May 1987.
- [Mackworth 77] A.K. Mackworth.
Consistency in Networks of Relations.
Artificial Intelligence 8(1):99-118, 1977.
- [Montanari 74] Montanari, U.
Networks of Constraints.
In *Proceedings IFIP Congress*, pages 727-732. 1974.
- [Muscettola 87] Nicola Muscettola, and Stephen Smith.
A Probabilistic Framework for Resource-Constrained Multi-Agent Planning.
In *Proceedings of the Tenth International Conference on Artificial
Intelligence*, pages 1063-1066. 1987.
- [Nadel 86a] B.A. Nadel.
The General Consistent Labeling (or Constraint Satisfaction) Problem.
Technical Report DCS-TR-170, Department of Computer Science, Laboratory
for Computer Research, Rutgers University, New Brunswick, NJ 08903,
1986.
- [Nadel 86b] B.A. Nadel.
*Three Constraint Satisfaction Algorithms and Their Complexities: Search-
Order Dependent and Effectively Instance-specific Results*.
Technical Report DCS-TR-171, Department of Computer Science, Laboratory
for Computer Research, Rutgers University, New Brunswick, NJ 08903,
1986.
- [Nadel 86c] B.A. Nadel.
Theory-based Search-order Selection for Constraint Satisfaction Problems.
Technical Report DCS-TR-183, Department of Computer Science, Laboratory
for Computer Research, Rutgers University, New Brunswick, NJ 08903,
1986.
- [Newell 69] Newell, A.
Heuristic Programming: Ill-Structured Problems.
Progress in Operations Research 3:360-414, 1969.

- [Newell&Simon 56] Newell, A., and Simon H. A.
The Logic Theory Machine: A Complex Information Processing System.
IRE Transactions on Information Theory IT-2 3:61-79, 1956.
- [Newell&Simon 63] Newell, A. and Simon, H.A.
Computers and Thought: GPS: A Program that Simulates Human Thought.
McGraw-Hill Co., New York, 1963.
- [Newell&Simon 72] Newell, A. and Simon, H.A.
Human Problem Solving.
Prentice Hall, 1972.
- [Nudel 83] Bernard Nudel.
Consistent-Labeling Problems and their Algorithms: Expected-Complexities
and Theory-Based Heuristics.
Artificial Intelligence 21:135-178, 1983.
- [Ow 84] Peng Si Ow.
Heuristic Knowledge and Search for Scheduling.
Technical Report, Robotics Institute, Carnegie Mellon University, Pittsburgh,
PA 15213, 1984.
(Ph.D. Thesis).
- [Purdom 83] Paul W. Purdom, Jr.
Search Rearrangement Backtracking and Polynomial Average Time.
Artificial Intelligence 21:117-133, 1983.
- [Rit 86] Jean-Francois Rit.
Propagating Temporal Constraints for Scheduling.
In *Proceedings of the Sixth National Conference on Artificial Intelligence*,
pages 383-388. 1986.
- [Sacerdoti 74] E.D. Sacerdoti.
Planning in a Hierarchy of Abstraction Spaces.
Artificial Intelligence 5(2):111-135, 1974.
- [Sadeh&Fox 88] N. Sadeh and M.S. Fox.
Preference Propagation in Temporal/Capacity Constraint Graphs.
Technical Report CMU-CS-88-193, Computer Science Department, Carnegie
Mellon University, Pittsburgh, PA 15213, 1988.
Also appears as Robotics Institute technical report CMU-RI-TR-89-2.
- [Smith 83] Stephen F. Smith.
Exploiting Temporal Knowledge to Organize Constraints.
Technical Report, Robotics Institute, Carnegie Mellon University, Pittsburgh,
PA 15213, 1983.

- [Smith 85] Stephen F. Smith and Peng Si Ow.
The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks.
In *Proceedings of the Ninth International Conference on Artificial Intelligence*, pages 1013-1015. 1985.
- [Smith, Fox &Ow 86] S. Smith, M. Fox, and P.S. Ow.
Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems.
AI Magazine 7(4):45-61, Fall, 1986.
- [Stefik 81] Mark Stefik.
Planning with Constraints (MOLGEN: Part 1).
Artificial Intelligence 16():111-140, 1981.
- [Stone 86] Harold S. Stone and Paolo Sipala.
The average complexity of depth-first search with backtracking and cutoff.
IBM Journal of Research and Development 30(3):242-258, 1986.
- [Stroud 71] A. H. Stroud.
Series in Automatic Computation: Approximate Calculations of Multiple Integrals.
Prentice Hall, 1971.
- [Thomas 83] Thomas and Finney.
Calculus and Analytic Geometry, Sixth Edition.
Addison Wesley, 1983.
- [Tsang 87] Edward P. K. Tsang.
The Consistent Labeling Problem in Temporal Reasoning.
In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 251-255. 1987.
- [Valdes-Perez 87] Raul E. Valdes-Perez.
The Satisfiability of Temporal Constraint Networks.
In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 256-260. 1987.
- [Vere 83] Stephen Vere.
Planning in Time: Windows and Durations for Activities and Goals.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5"(3):246-267, 1983.
- [Vilain 86] Marc Vilain and Henry Kautz.
Constraint Propagation Algorithms for Temporal Reasoning.
In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 377-382. 1986.

DISTRIBUTION LIST

addresses	number of copies
Northrup Fowler III RADC/COES	20
RADC/DOVL GRIFFISS AFB NY 13441	1
RADC/IMP GRIFFISS AFB NY 13441	2
ADMINISTRATOR DEF TECH INF CTR ATTN: DTIC-FDA CAMERON STA BG 5 ALEXANDRIA VA 22304-6145	5
RADC/COTD BLDG 3, ROOM 16 GRIFFISS AFB NY 13441-5700	1
HQ USAF/SCTT Pentagon Wash DC 20330-5190	1
DIRECTOR DMAHTC ATTN: SDSIM Wash DC 20315-0030	1
Director, Info Systems OASD (C3I) Rm 3E187 Pentagon Wash DC 20301-3040	1
Naval Warfare Assessment Center Attn: GIDEP Operations Center Code 3UG1 (E. Richards) Corona CA 91720	1

HQ AFSC/XTKT
ANDREWS AFB DC 20334-5001

1

HQ AFSC/XRK
ANDREWS AFB MD 20334-500

1

HQ SAC/SCPT
OFFUTT AFB NE 68113-5001

1

DTESA/RQE
ATTN: LARRY G. MCMANUS
KIRTLAND AFB NM 87117-5000

1

HQ TAC/DRIY
Attn: Mr. Westerman
Langley AFB VA 23665-5001

1

HQ TAC/DRCA
LANGLEY AFB VA 23665-5001

1

ASD/ENEMS
Wright-Patterson AFB OH 45433-6503

2

ASD/AFALC/AXAE
Attn: W. H. Dungey
Wright-Patterson AFB OH 45433-6533

1

AFIT/LDEE
BUILDING 640, AREA B
WRIGHT-PATTERSON AFB OH 45433-6583

1

WRDC/MLPC
WRIGHT-PATTERSON AFB OH 45433-6533

1

AAMRL/HE
WRIGHT-PATTERSON AFB OH 45433-6575

1

Air Force Human Resources Laboratory
Technical Documents Center
AFHRL/LRS-TDC
Wright-Patterson AFB OH 45433

1

2750 ABW/SSLT
Bldg 262
Post 11S
Wright-Patterson AFB OH 454433

1

AUL/LSE
MAXWELL AFB AL 36112-5564

1

Defense Communications Engineering Ctr
Technical Library
1860 Wiehle Avenue
Reston VA 22090-5500

1

COMMAND CONTROL AND COMMUNICATIONS DIV
DEVELOPMENT CENTER
MARINE CORPS DEVELOPMENT & EDUCATION COMMAND
ATTN: CCDE DICA
QUANTICO VA 22134-5080

2

AFLMC/LGY
ATTN: CH, SYS ENGR DIV
GUNTER AFS AL 36114

1

U.S. Army Strategic Defense Command
Attn: DASD-H-MPL
P.O. Box 1500
Huntsville AL 35807-3801

1

COMMANDING OFFICER
NAVAL AVIONICS CENTER
LIBRARY - D/765
INDIANAPOLIS IN 46219-2189

1

COMMANDING OFFICER
NAVAL TRAINING SYSTEMS CENTER
TECHNICAL INFORMATION CENTER
BUILDING 2068
ORLANDO FL 32813-710C

1

COMMANDER
NAVAL OCEAN SYSTEMS CENTER
ATTN: TECHNICAL LIBRARY, CODE 9642B
SAN DIEGO CA 92152-5000

1

COMMANDER (CODE 3433)
ATTN: TECHNICAL LIBRARY
NAVAL WEAPONS CENTER
CHINA LAKE, CALIFORNIA 93555-6001

1

SUPERINTENDENT (CODE 1424)
NAVAL POST GRADUATE SCHOOL
MONTEREY CA 93943-5000

1

COMMANDING OFFICER
NAVAL RESEARCH LABORATORY
ATTN: CODE 2627
WASHINGTON DC 20375-5000

2

SPACE & NAVAL WARFARE SYSTEMS COMMAND
PMW 153-3DP
ATTN: R. SAVARESE
WASHINGTON DC 20363-5100

1

CDR, U.S. ARMY MISSILE COMMAND
REDSTONE SCIENTIFIC INFORMATION CENTER
ATTN: AMSMI-RD-CS-R (DOCUMENTS)
REDSTONE ARSENAL AL 35898-5241

2

Advisory Group on Electron Devices
Hammond John/Technical Info Coordinator
201 Varick Street, Suite 1140
New York NY 10014

2

UNIVERSITY OF CALIFORNIA/LOS ALAMOS
NATIONAL LABORATORY
ATTN: DAN BACA/REPORT LIBRARIAN
P.O. BOX 1663, MS-P364
LOS ALAMOS NM 87545

1

RAND CORPORATION THE/LIBRARY
HELPER DORIS S/HEAD TECH SVCS
P.O. BOX 2138
SANTA MONICA CA 90406-2138

1

AEDC LIBRARY (TECH REPORTS FILE)
MS-100
ARNOLD AFS TN 37389-9998

1

USAG
Attn: ASH-PCA-CRT
Ft Huachuca AZ 85613-6000

1

JTFPO-TD
Attn: Director/Advanced Technology
1500 Planning Research Drive
McLean VA 22102-5099

1

AFWC/ESRI
SAN ANTONIO TX 78243-5000

3

485 EIG/EIER (DMO)
GRIFFISS AFB NY 13441-6348

1

ESD/AVS
ATTN: ADV SYS DEV
HANSCOM AFB MA 01731-5000

1

ESD/ICP
HANSCOM AFB MA 01731-5000

1

ESD/AVSE
BLOG 1704
HANSCOM AFB MA 01731-5000

2

HQ ESD SYS-2
HANSCOM AFB MA 01731-5000

1

The Software Engineering Institute Attn: Major Dan Burton, USAF Joint Program Office Carnegie Mellon University Pittsburgh PA 15213-3890	1
DIRECTOR NSA/CSS ATTN: TS13/TDL (DAVID MARJAPUM) FORT GEORGE G MEADE MD 20755-6000	1
DIRECTOR NSA/CSS ATTN: R24 FORT GEORGE G MEADE MD 20755-6000	1
DIRECTOR NSA/CSS ATTN: R21 9800 SAVAGE ROAD FORT GEORGE G MEASDE MD 20755-6000	1
DIRECTOR NSA/CSS ATTN: R5 FORT GEORGE G MEADE MD 20755-6000	1
DIRECTOR NSA/CSS ATTN: R8 FORT GEORGE G MEADE MD 20755-6000	1
DIRECTOR NSA/CSS ATTN: S21 FORT GEORGE G MEADE MD 20755-6000	1
DIRECTOR NSA/CSS ATTN: W3 FORT GEORGE G MEADE MD 20755-6000	1
DoD COMPUTER SECURITY CENTER ATTN: C4/TIC 9800 SAVAGE ROAD FORT GEORGE G MEADE MD 20755-6000	1

ESD-MITRE Software Center Library c/o Ms J.A. Clapp MITRE Corp, D-70 Burlington Road Bedford, MA 01730	2
Software Engineering Institute Tech Lib. ATTN: Korola Fuchs Carnegie--Mellon University Pittsburgh, PA 15232	2
Dr. Edward A. Feigenbaum Knowledge Systems Laboratory Stanford University 701 Welch Road, Bldg C Palo Alto, CA 94304	1
Dr. Steven Vere Lockheed AI Center, 90-06/259 Lockheed Research & Development Division 3251 Hanover St. Palo Alto, CA 94304-1187	1
Dr. Austin Tate, Director AIAI University of Edinburgh 80 South Bridge Edinburgh, Scotland EH1 1HN	1
Miroslav Benda Boeing P.O. Box 24346, M/S 7L-64 Seattle, WA 98124	1
Dr. Saul Amarel Department of Computer Science Busch Campus Rutgers University New Brunswick, NJ 08903	1
Charles F. Schmidt Department of Computer Science Busch Campus Rutgers University New Brunswick, NJ 08903	1
Mr Robert Drazcovich Advanced Decision Systems 1500 Plymouth St. Mountain View, CA 94043-1230	1

Dr Brian P. McCune
Advanced Decision Systems
1500 Plymouth St.
Mountain View, CA 94043-1230

1

Andrew S. Cromarty
Advanced Decision Systems
1500 Plymouth St.
Mountain View, CA 94043-1230

1

Dr. Drew McDermott
Dept of Computer Science
Yale University
P.O. Box 2158 Yale Station
New Haven, CT 06520

1

W. A. Frawley
GTE Labs
40 Sylvan Road
Waltham, MA 02254

1

Dr. Randell Schumaker
Code 7510
NRL
4555 Overlook Ave, SW
Washington, DC 20375

1

Robert Lawler
Boeing Computer Services
Advanced Technology Applications Division
P.O. Box 24346
Seattle, WA 98124-0346

1

R. Bruce Roberts
Bolt Beranek and Newman Inc.
Department of Artificial Intelligence
10 Moulton Street
Cambridge, MA 02238

1

Dr Eugene Charniak
Brown University
Box 1910
Providence, RI 02912

1

Dr Jaime Carbone
Carnegie-Mellon University
Computer Science Department
Schenley Park
Pittsburgh, PA 15213

1

Dr Mark Fox Carnegie-Mellon University Intelligent Systems Laboratory The Robotics Institute Pittsburgh, PA 15213	1
Dr John Hoptcroft Cornell University Computer Science Department Upson Hall Ithaca, NY 14853	1
Lt C. Robert Simpson DARPA/ISTO 1400 Wilson Blvd Arlington, VA 22209-2389	2
Dr Donald W. Lovelanc Duke University Computer Science Department Durham, NC 27706	1
Dr Perry W. Thorndyke FMC Corporation Central Engineering Laboratories 1205 Coleman Avenue, Box 580 Santa Clara, CA 95052	1
Dr Piero P. Bonissone General Electric Company Corporate Research and Development 1 River Road, Building 37-567 Schenectady, NY 12345	1
Thomas E. Cheatham Harvard University Aiken Computation Laboratory 33 Oxford Street Cambridge, MA 02138	1
John R. Beane Honeywell, Inc. Systems & Research Center MN17-2346 2600 Ridgway Parkway NE Minneapolis, MN 55413	1
Robert C. Schrag Honeywell, Inc. Systems & Research Center MN17-2346 2600 Ridgway Parkway NE Minneapolis, MN 55413	1
Dr. Philip Klahr Inference Corporation 5300 West Century Boulevard Los Angeles, CA 90045	1

Dr Paul Morris IntelliCorp 1975 El Camino Real West Mountain View, CA 94040-2216	1
Dr Thomas P. Kehler IntelliCorp 1975 El Camino Real West Mountain View, CA 94040-2216	1
Dr Gerard T. Capraro Kaman Sciences Corporation 258 Genesee Street Utica, NY 13502	1
Dr Cordell Green Kestrel Institute 1801 Page Mill Road Palo Alto, CA 94304	1
Dr. John Lemmer Knowledge Systems Concepts, Inc. 225 North Washington Street P. O. Box 508 Rome, NY 13440	1
Dr Christine A. Montgomery Logicon, Inc. Operating Systems Division 21031 Ventura Boulevard Woodland Hills, CA 91364	1
Richard H. Hill Microelectronics and Computer Technology Corp. Echelon Building #1, Suite 200 9430 Research Boulevard Austin, TX 78759	1
Dr Randall Davis MIT Artificial Intelligence Laboratory Room NE43-801A 545 Technology Square Cambridge, MA 02139-1986	1
Dr Charles Rich MIT Artificial Intelligence Laboratory Room NE43-839 545 Technology Square Cambridge, MA 02139-1986	1

Dr Patrick Winston MIT Artificial Intelligence Laboratory Room NE43-816 545 Technology Square Cambridge, MA 02139-1986	1
Dr Ramesh S. Patil MIT Laboratory for Computer Science Room NE43-316 545 Technology Square Cambridge, MA 02139-1986	1
Dr Peter Szolovits MIT Laboratory for Computer Science Clinical Decision Making Group 545 Technology Square Cambridge, MA 02139-1986	1
Dr J.A. Robinson University Professor Syracuse University Syracuse, N.Y. 13244	1
Dr B. Chandrasekaran Ohio State University Depart. of Computer and Information Sciences 2036 Neil Avenue Columbus, OH 43210	1
Dr John Josephson Ohio State University Depart. of Computer and Information Sciences 2036 Neil Avenue Columbus, OH 43210	1
Dr Jude E. Franklin Planning Research Corporation Research and Development Technology Division 1500 Planning Research Drive McLean, VA 22102	1
Sanjai Narain The Rand Corporation Information Sciences Department 1700 Main Street Santa Monica, CA 90406	1
Dr Harvey Rhody RIT Research Corporation 75 Hightower Road Rochester, New York 14623	1

Dr Casimir A. Kulikowski Rutgers University Department of Computer Science Hill Center, Busch Campus New Brunswick, NJ 08903	1
Dr Tom Mitchell Rutgers University Department of Computer Science Hill Center, Busch Campus New Brunswick, NJ 08903	1
Dr Sholom Weiss Rutgers University Department of Computer Science Hill Center, Busch Campus New Brunswick, NJ 08903	1
Dr Jay M. Tenenbaum Schlumberger Palo Alto Res Center 3540 Hillview Ave. Palo Alto, CA 94304	1
Dr David Barstow Schlumberger-Doll Research Center Old Quarry Road Ridgefield, CT 06877-4108	1
Dr Elaine Kant Schlumberger-Doll Research Center Old Quarry Road Ridgefield, CT 06877-4108	1
Dr Richard J. Waldinger SRI International Artificial Intelligence Center 355 Ravenswood Avenue Menlo Park, CA 94025-3493	1
Dr Jan Aikens Computer Science Department Stanford University Margaret Jacks Hall Stanford, CA 94305	1
Dr Zohar Manna Computer Science Department Stanford University Margaret Jacks Hall Stanford, CA 94305	1
Dr Nils J. Nilsson, Chairman Computer Science Department Stanford University Margaret Jacks Hall Stanford, CA 94305	1

Dr Nelleke Aiello Stanford University Heuristic Programming Project 701 Welch Road, Building C Palo Alto, CA 94304	1
Dr Harold Brown Stanford University Heuristic Programming Project 701 Welch Road, Building C Palo Alto, CA 94304	1
Dr Bruce G. Buchanan Stanford University Heuristic Programming Project 701 Welch Road, Building C Palo Alto, CA 94304	1
Dr Robert Engelmores Stanford University Heuristic Programming Project 701 Welch Road, Building C Palo Alto, CA 94304	1
Dr Larry Fagan Stanford University Heuristic Programming Project 701 Welch Road, Building C Palo Alto, CA 94304	1
Dr Michael R. Genesereth Stanford University Heuristic Programming Project 701 Welch Road, Building C Palo Alto, CA 94304	1
Dr Barbara Hayes-Roth Stanford University Heuristic Programming Project 701 Welch Road, Building C Palo Alto, CA 94304	1
Dr H. Penny Nii Stanford University Heuristic Programming Project 701 Welch Road, Building C Palo Alto, CA 94304	1
Dr Edward H. Shortliffe Stanford University Medical Center Division of General Internal Medicine Medical Computer Science TC-135 Stanford, CA 94305	1
Dr Stuart C. Shapiro SUNY/Buffalo Computer Science Department 226 Bell Hall Buffalo, NY 14260	1

Dr Sargur N. Srihari SUNY/Buttalo Computer Science Department 226 Bell Hall Buffalo, NY 14260	1
Dr Richard O. Duda Syntelligence 1000 Hamlin Court Sunnyvale, CA 94088	1
Dr Peter E. Hart Syntelligence 1000 Hamlin Court Sunnyvale, CA 94088	1
Dr Frederick Hayes-Roth Teknowledge, Inc. 1850 Embarcadero Palo Alto, CA 94301	1
Bruce Bullock Teknowledge Federal Systems 501 Marin Street, #214 Thousand Oaks, CA 91360	1
Gary Edwards Teknowledge Federal Systems 501 Marin Street, #214 Thousand Oaks, CA 91360	1
Dr Roger Bate, Director Texas Instruments Central Research Labs Computer Science Laboratory P.O. Box 226015, MS 238 Dallas, TX 75266	1
Dr Ed Taylor TRW Defense & Space Group Building R2/2094 One Space Park Redondo Beach, CA 90278	1
Dr Paul Cohen University of Massachusetts Computer & Information Science Department Amherst, MA 01003	1

Dr W. Bruce Croft University of Massachusetts Computer & Information Science Department Amherst, MA 01003	1
Dr Victor R. Lesser University of Massachusetts Computer & Information Science Department Amherst, MA 01003	1
Dr Harry E. Pople University of Pittsburgh Decision Systems Laboratory 1360 Scaife Hall Pittsburgh, PA 15261	1
Dr James F. Allen University of Rochester Department of Computer Science Rochester, NY 14627	1
Dr Robert M. Balzer University of Southern California Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292-6695	1
Dr Lee Erman Teknowledge, Inc 1850 Embarcadero Palo Alto, CA 94301	1
Dr Ronald Ohlander University of Southern California Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292-6695	1
Dr William R. Swartout University of Southern California Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292-6695	1
Dr J. C. Brown University of Texas at Austin Department of Computer Sciences Austin, TX 78712-1188	1

Advanced Computer Architectures/Systems
MCC
3500 West Balcones Center Drive
Austin, Texas 78759

Dr Benjamin Kuipers 1
University of Texas at Austin
Department of Computer Sciences
T. S. Painter Hall 3.28
Austin, TX 78712-1188

Dr Bruce Porter 1
University of Texas at Austin
Department of Computer Sciences
Austin, TX 78712-1188

Dr Daniel G. Bobrow 1
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Dr Jonan de Kleer 1
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Dr Mark Stefik 1
Xerox Corp
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Dr Christopher Riesbeck 1
Yale Univ (Computer Science)
P.O. Box 2158, Yale Station
New Haven, CT 06520

Mark Burstein 1
BBN Laboratories, Inc.
10 Mcuton Street
Cambridge, MA 02238

David Chapman 1
MIT AI Laboratory
545 Technology Square
Cambridge, MA 02139

Thomas Dean 1
Box 191U
Brown University
Computer Science Department
Providence, RI 02912

Peter Friedland 1
NASA Ames Research Center
RIA:244-17
Moffett Field, CA 94035

Mike Georgeff 1
Australian AI Institute
1 Grattan St., Carlton
Victoria 3053
Australia

Kristian Hammond 1
The University of Chicago
1100 East 58th Street
Ryerson 152
Chicago, IL 60637

Richard Korf 1
UCLA
3532H Boelter Hall
Computer Science Department
Los Angeles, CA 90024

Paul Lehner 1
George Mason University
Info Technology & Engineering
4400 University Drive
Fairfax, VA 22033

Ted Linden 1
ADS
1500 Plymouth St.
Mountain View, CA 94043-1230

Tomas Lozano-Perez 1
MIT AI Laboratory
545 Technology Square
Cambridge, MA 02139

Matt Ginsberg 1
Stanford University
Computer Science Department
Stanford, CA 94305

Steven Smith c/o Patty Hodgson 1
Carnegie-Mellon University
Robotics Institute
Schenley Park
Pittsburgh, PA 15213

Raj Wall Texas Instruments, Inc. Artificial Intelligence Laboratory P.O. Box 655474, M/S 238 Dallas, TX 75265	1
Ben Wise Dartmouth College Thayer School of Engineering Hanover, NH 03755	1
Bolt Beranek Newman Laboratories ATTN: Dr. Ed. Walker CASES Program Manager 10 Moulton St. Cambridge, MA 02238	1
Elliot Soloway University of Michigan Dept. of Elect. Eng. & Computer Science 1301 Beale Ave. 3300 EECS Building Ann Arbor, MI 48109	1
Judea Pearl 4751 Boelter Hall UCLA Los Angeles, CA 90024	1
Dr. William Clancy Institute for Research Learning 3555 Coyote Hill Rd. Palo Alto, CA 94304	1
Prof Eric Sandewall Dept of Computer Science Linköping University 58183 Linköping Sweden	1
Yoav Shoham Dept of Computer Science Stanford University M/S 2140 Stanford, CA 94305	1
Ken Forbus Qual Reasoning Group Univ. of Illinois 1304 W. Springfield Ave. Urbana, IL 61801	1

Rick Alterman Computer Science Department Ford Hall Brandeis University Waltham, MA 02254	1
Mark Drummond NASA Ames Research Center Mail Stop: 244-17 Moffett Field, CA 94035	1
Richard Fikes Price Waterhouse Technology Center 68 Willow Road Menlo Park, CA 94025	1
R. James Firby PO Box 6636 Yale Station Yale University New Haven, CT 06520	1
Jim Hendler Computer Science Department University of Maryland College Park, MD 20742	1
Francois F. Ingrand SRI AI Center SRI International 333 Ravenswood Ave Menlo Park, CA 94025	1
Leslie Kaelbling Teleos Research 576 Middlefield Rd. Palo Alto, CA 94301	1
Henry Kautz AT&T Bell Labs 600 Mountain Ave, Room 3C-402A Murray Hill, NJ 07974	1
Amy L. Lansky AI Center SRI International 333 Ravenswood Ave Menlo Park, CA 94025	1

Dana S. Nau
Computer Science Dept.
University of Maryland
College Park, MD 20742

1

Ray Perrault
SRI AI Center
333 Ravenswood Ave.
Menlo Park, CA 94025

1

Stan Rosenschein
Telecs Research
576 Middlefield Road
Palo Alto, CA 94301

1

Jim Schmolze
Dept. of Computer Science
Tufts University
Medford, MA 02155

1

Candy Sicner
12 Aiken Labs
Harvard University
Cambridge, MA 02138

1

David E. Smith
Rockwell Palo Alto Science Center
444 High St.
Palo Alto, CA 94301

1

Michael Fehling
Rockwell Palo Alto Science Center
444 High Street
Palo Alto, CA 94301

1

N.S. Sridharan
FMC Corporation
Central Engin. Lab, Box 580
1205 Coleman Ave.
Santa Clara, CA 95052

1

Katia Sycara Robotics Institute Carnegie Mellon University Pittsburgh PA, 15213	2
Josh Tenenber Dept. of Computer Science University of Rocheszter Rochester, NY 14627	1
Monte Zweben NASA AMES Research Center Mail Stop 244-17 Moffett Field, CA 94035	1
Alice M. Agogino Dept of Mechanical Engineering 5136 Etcheverry Hall University of California, Berkeley Berekley, CA 94720	1
Robert Beaton Draper Laboratory Cambridge, MA 02139	1
Glen Castore Honeywell Systems and Research Center 3660 Technology Dr. Minneapolis, MN 55418	1
Norman H. Chang Dept. of Elect. Eng. and Com. Science University of California, Berkeley Berkeley, CA 94704	1
Bruce D'Ambrosio Dept of Computer Science Oregon State University Corvallis, OR 97331-4602	1

1

Karen Johnson
Texas Instruments
PO Box 600246
Mail Stop 3645

1

Eric Horvitz
Knowledge Systems Laboratory
Stanford University
701 Welch Rd., Building C
Palo Alto, CA 94394

1

Thomas J. Laffey
Lockheed Missile & Space Co.
0/92-10 8/257
3251 Hanover Street
Palo Alto, CA 94304

1

A. Gabrielian
Thomson-CSF, Inc
630 Hansen Way, Suite 250
Palo Alto, CA 94304